Win-GRAF User Manual

ICP DAS CO., LTD. would like to congratulate you own your purchase of our Win-GRAF PACs. The ease to integration of the controller system and the power of the Win-GRAF software program combine to make a powerful, yet inexpensive industrial process control system.

Win-GRAF PAC (Programming Automation Controller) Series of ICP DAS includes:

ViewPAC-1000:	**VP-1238-CE7**
ViewPAC-2000:	**VP-2208-CE7, VP-6208-CE7, VP-2238-CE7, VP-6238-CE7**
ViewPAC-4000:	**VP-4208-CE7, VP-4238-CE7**
WinPAC-5000:	**WP-5238-CE7**
WinPAC-8000:	**WP-8148, WP-8448, WP-8848**
	WP-8128-CE7, WP-8428-CE7, WP-8828-CE7
XPAC-8000-CE6	**XP-8048-CE6, XP-8348-CE6, XP-8748-CE6**
	XP-8038-CE6, XP-8138-CE6, XP-8338-CE6, XP-8738-CE6

Legal Liability

ICP DAS assumes no liability for any damage resulting from the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Trademarks & Copyright

The version number and release date of this document are listed at the bottom of each page.
Users can download the latest document on FTP:
http://ftp.icpdas.com/pub/cd/win-graf-workbench-cd/tutorials/

Technical Service

Win-GRAF Web site:
http://www.icpdas.com/root/product/solutions/softplc_based_on_pac/win-graf/win-graf.html

New Win-GRAF workbench, Lib and PAC driver:
http://www.icpdas.com/root/product/solutions/softplc_based_on_pac/win-graf/download/win-graf-driver.html

If you have any problems, please feel free to contact us. Email: service@icpdas.com.

Table of Contents

Chapter 1 Software Installation & Hardware Setting

1.1 Installing the Win-GRAF Workbench

Before installing the Win-GRAF Workbench, check the installation environment on your PC.

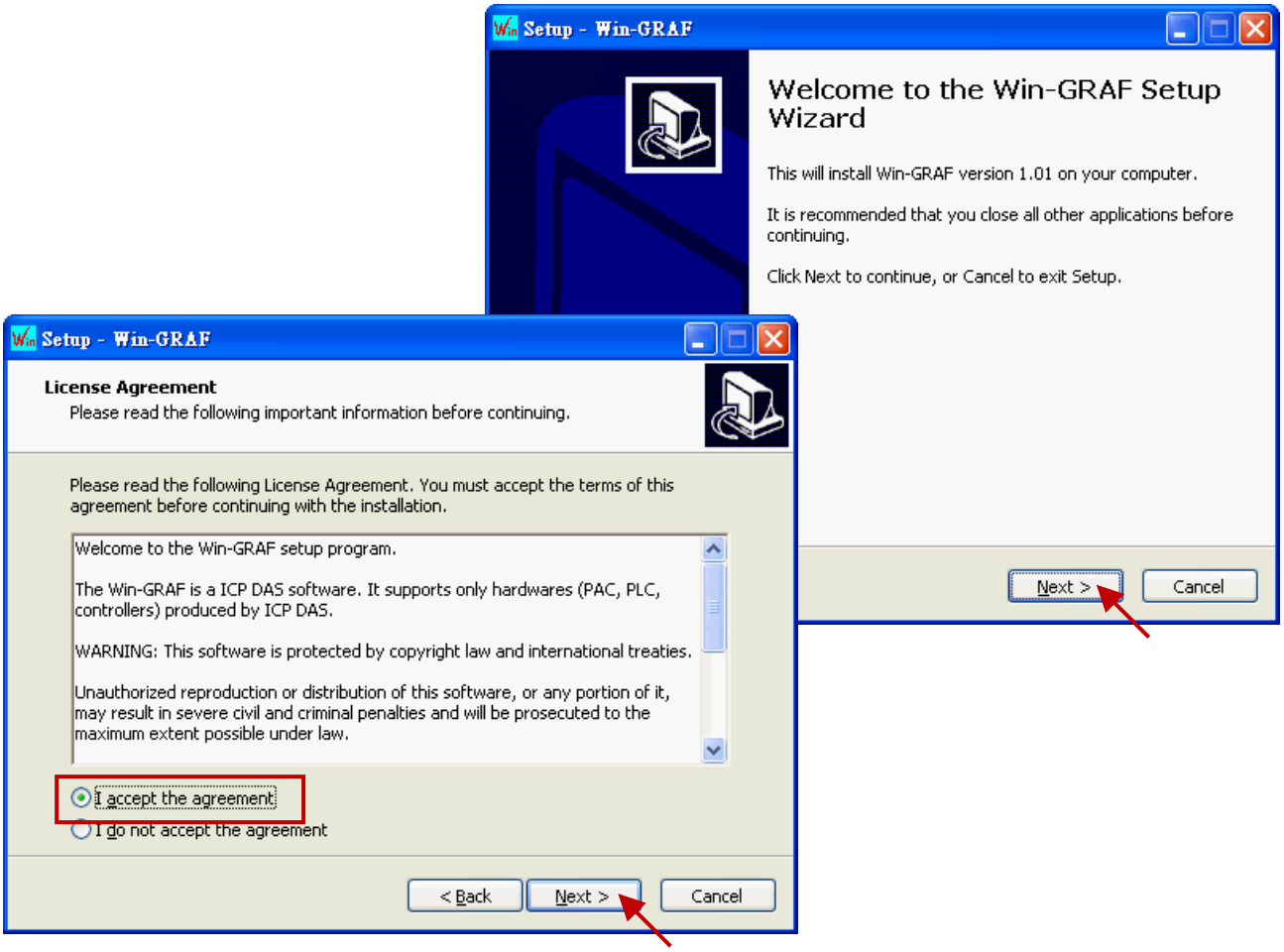
System requirements:

- **O.S.**: Windows XP, Windows Vista, Windows 7, Windows 8 (32-bits or 64-bits)
- **Microsoft .Net Framework 3.5** (Download it on the Microsoft web site: http://www.microsoft.com/zh-tw/download/details.aspx?id=22)
- **RAM:** 1 GB minimum (Recommended: 2 GB or more)
- **Available hard-disk space:** 200 MB minimum

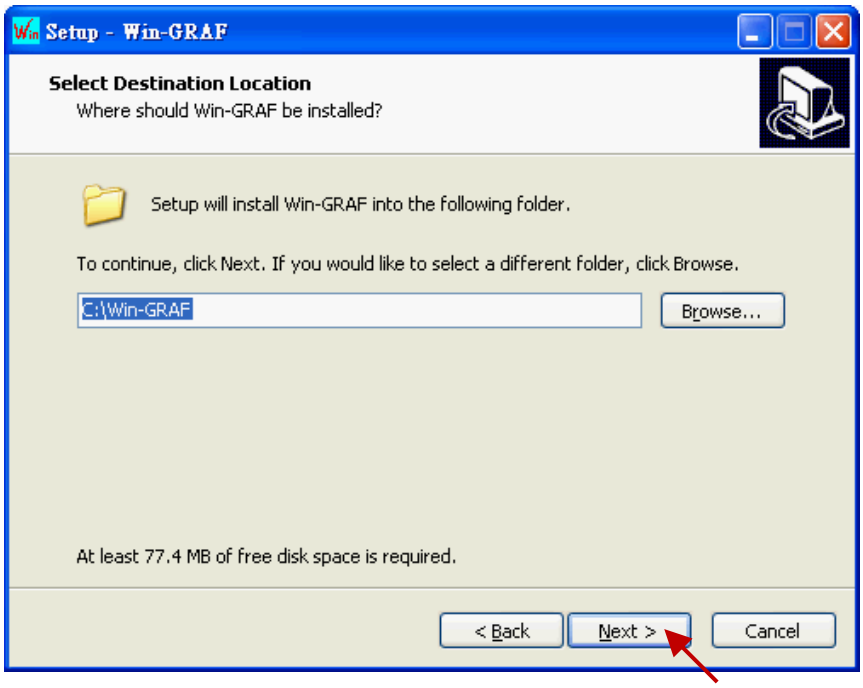
Installation Steps:

1. Double-click the "Win-GRAF-setup-ver-x.xx.exe" file in the Win-GRAF installation CD (or download the latest version of the Win-GRAF Workbench on the website: http://www.icpdas.com/root/product/solutions/softplc_based_on_pac/win-graf/download/win-graf-driver.html) to begin the process.

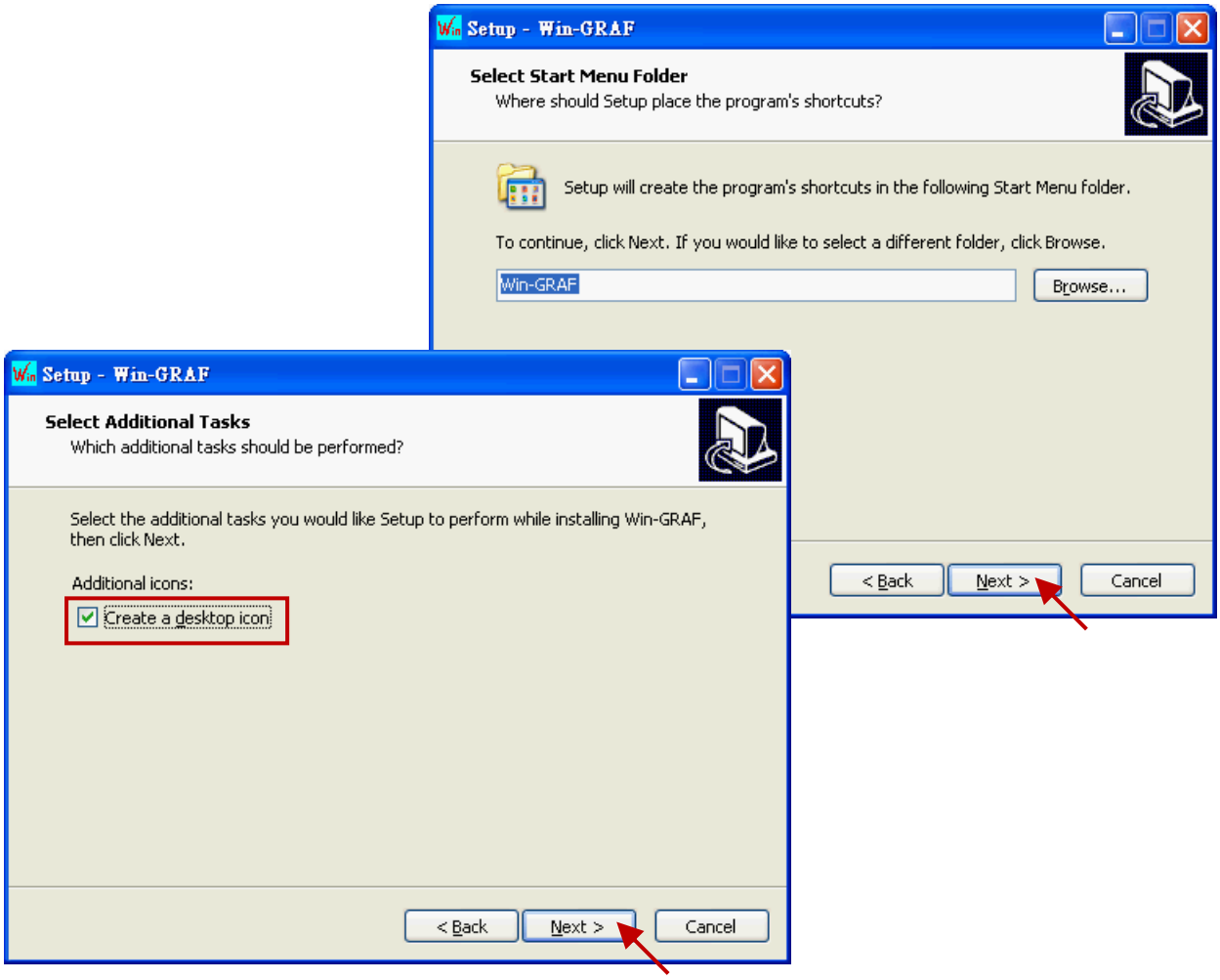
2. Click "Next" to continue and then select "I accept the agreement", then click "Next" to continue.



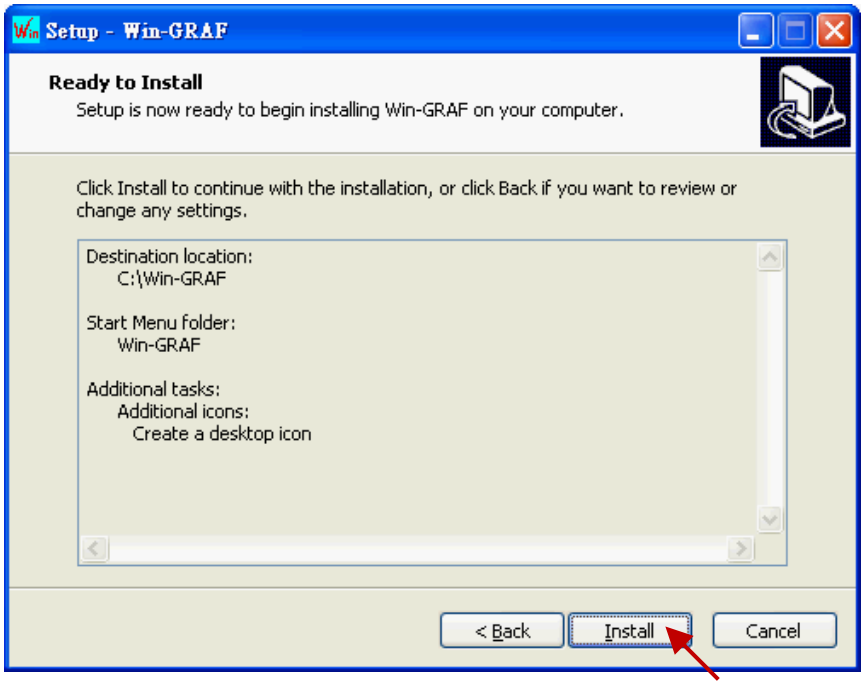
3. Recommend to use the default installation path (i.e., “C:\Win-GRAF”) and then click “Next” to continue.



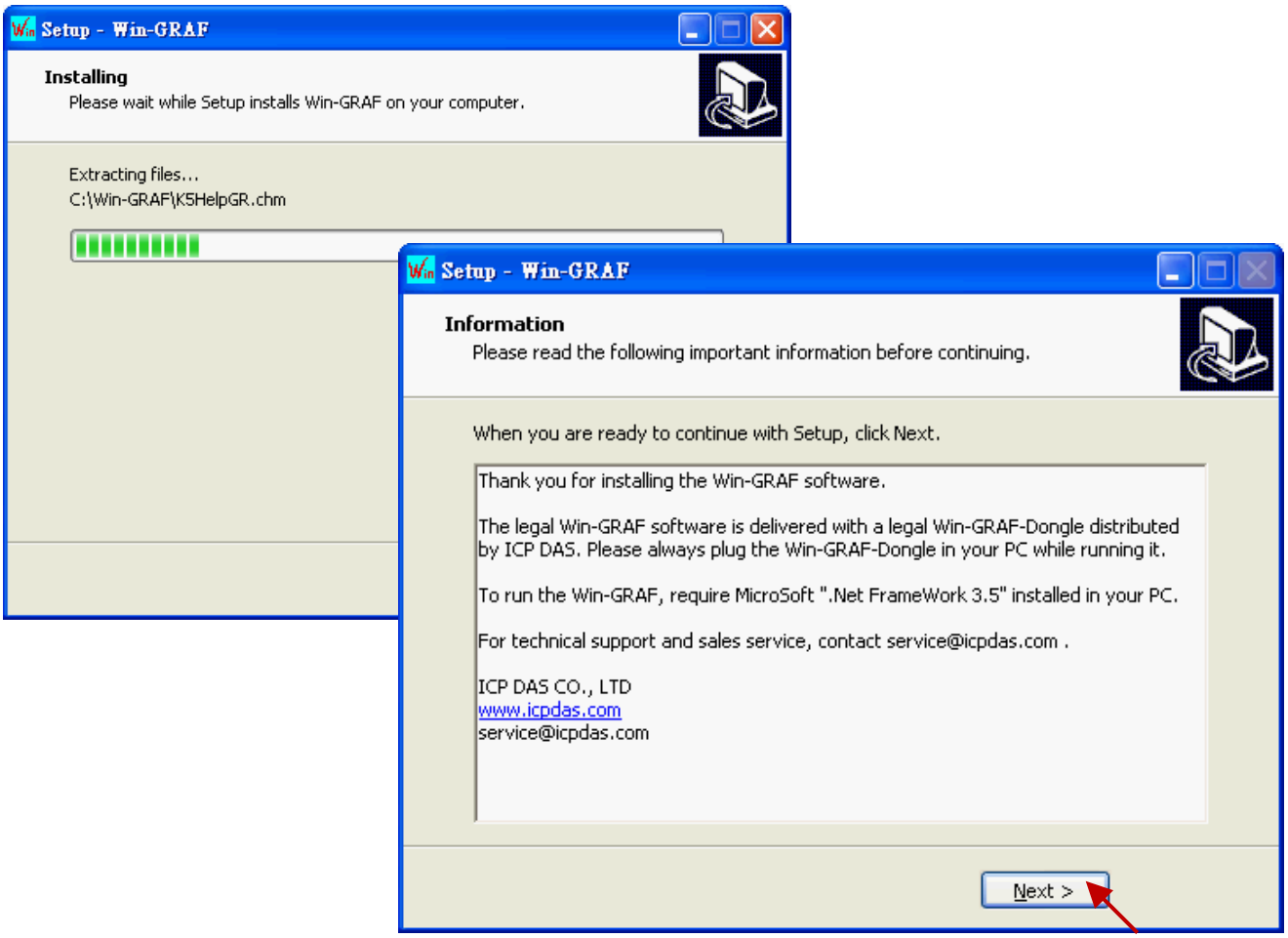
4. Click “Next” to add a “Win-GRAF” folder shortcut in the “Start” menu, and then select “Create a desktop icon” to add a desktop shortcut, then click “Next” to continue.



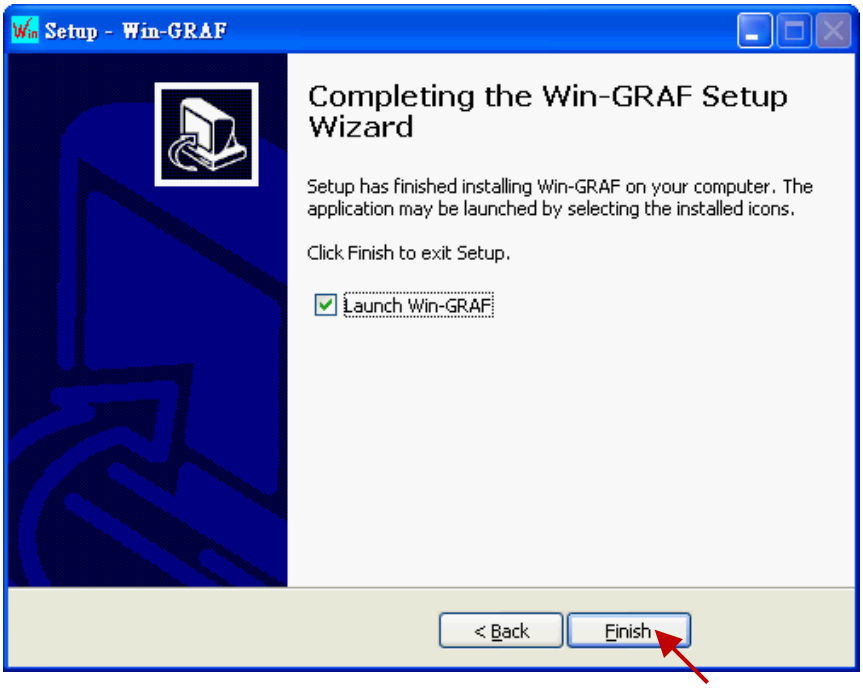
5. Click “Install” to begin installing the Win-GRAF Workbench.



6. Before the end of the installation, you will see a pop up window and it displays:
   a. The legal Win-GRAF Workbench is delivered with a legal Win-GRAF Dongle distributed by ICP DAS. Please always plug the Win-GRAF Dongle in your PC while running it.
   b. To run the Win-GRAF, require Microsoft “.Net FrameWork 3.5” installed in your PC.

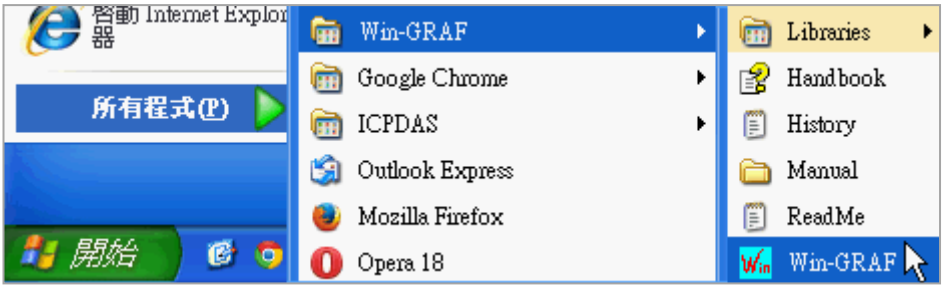


7. By now, you have completed the Win-GRAF installation, then click “Finish” to exit this window. (Select “Launch Win-GRAF” to auto-run the Win-GRAF after completing the setup. If there is no Win-GRAF Dongle in your PC, the Win-GRAF Workbench will run in Demo mode.)



1.2 Run the Win-GRAF Workbench

Before running the Win-GRAF Workbench, make sure the Win-GRAF Dongle is plugged into your PC. Without using a Win-GRAF Dongle, the Win-GRAF Workbench will run in Demo Mode. Open the Windows Start menu, click on “Win-GRAF” folder and “Win-GRAF” to open this software.



Description of the “Win-GRAF” folder:

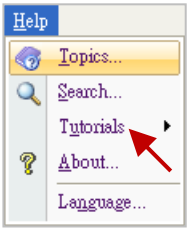
Libraries: For users to create their own function or modify an exist function.

Handbook: The manual details the software interface, programming environment, programming languages, and so on, provided by COPALP.
(Or click the [Help] > [Topics] from the Win-GRAF menu bar)

History: The modification history and features added of the Win-GRAF Workbench.

Manual: The Win-GRAF manual provided by ICP DAS.
(Or click the [Help] > [Tutorials] from the Win-GRAF menu bar, the manual is located in the path “C:\Win-GRAF\Tutorials”)

ReadMe: The notice for the Win-GRAF Workbench.



1.2.1 Win-GRAF Operating Mode

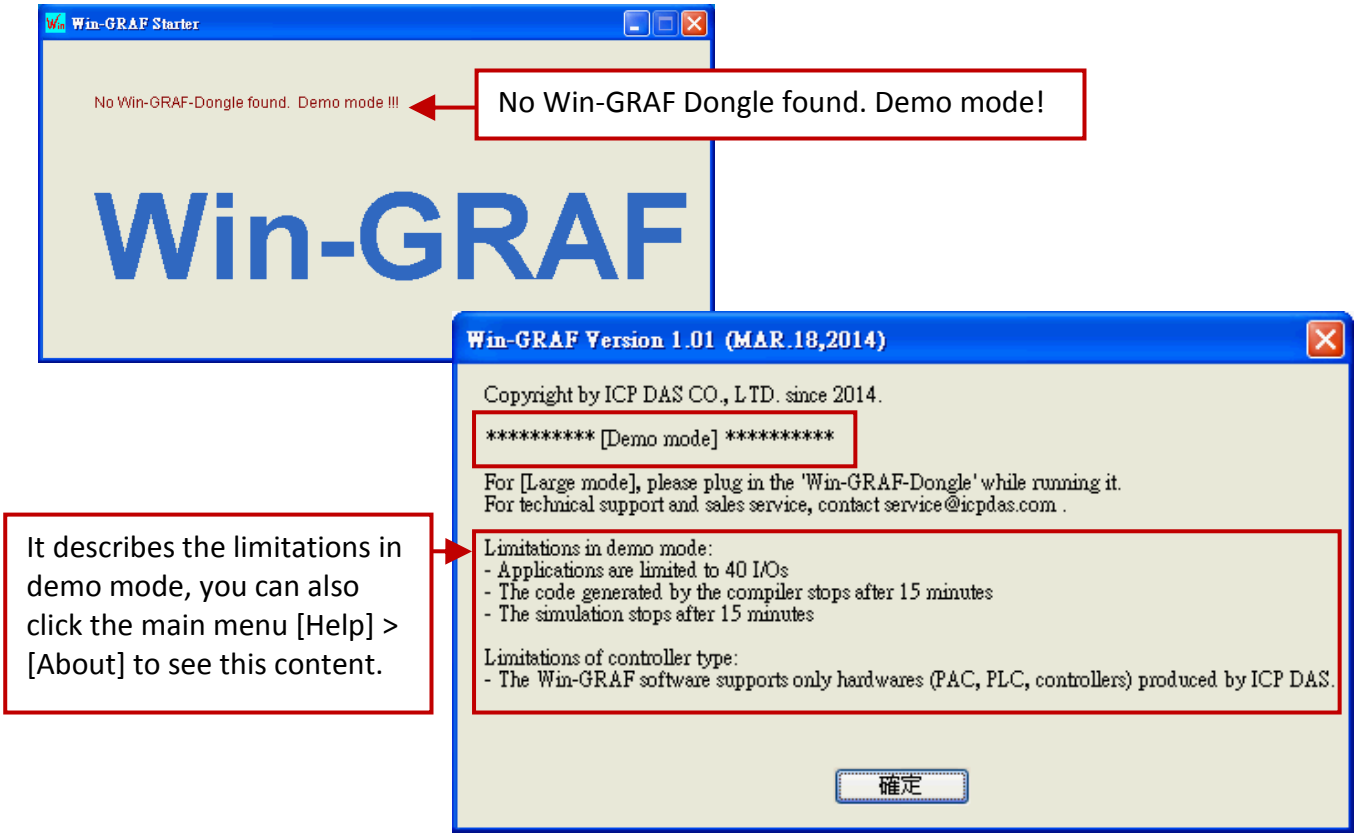
The Win-GRAF Workbench provides two operating modes:

"Demo Mode": Without using a Win-GRAF Dongle. The compiled Win-GRAF project can run for 15 minutes on the PAC. Once the time limit has expired, users must Stop/Start this project again and it only supports up to 40 I/O tags.

"Large Mode": Using a Win-GRAF Dongle. The project can run on the PAC without the time limit.

Demo Mode - Without using a Win-GRAF Dongle.

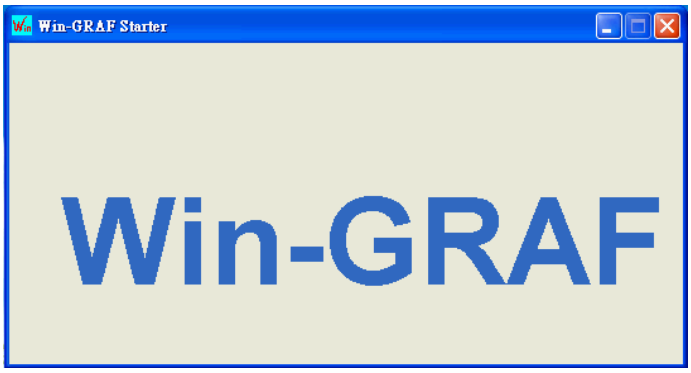
The start screen will show as below after running the Win-GRAF Workbench.



Note: If you install the Win-GRAF Dongle in the Demo Mode, you must close Win-GRAF Workbench and then start it again to make it become Large Mode.

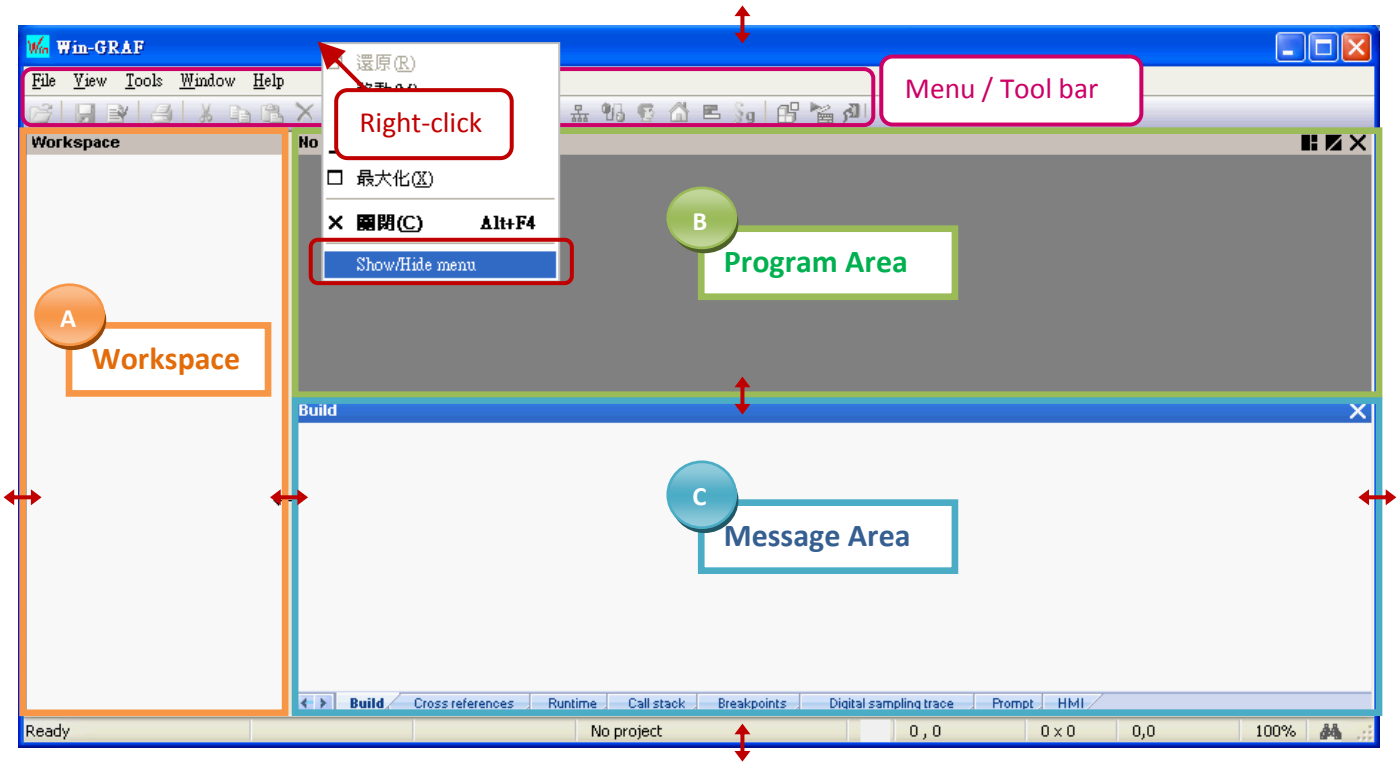
Large Mode - Using a Win-GRAF Dongle.

The start screen will show as below after running the Win-GRAF Workbench.



1.2.2 Win-GRAF Operating Environment

Run the Win-GRAF and then the main screen will show as below:



Note: Mouse right-click on the top of the Window to Show/Hide the menu bar.

A. The Workspace: It allows users to create project lists, and add/open the Win-GRAF program as well as the related settings. Moreover, the new project can be created by using a project template.

B. The Program Area: It used to show/edit the program and can be divided into more function area. (Refer the Section 2.2.1)

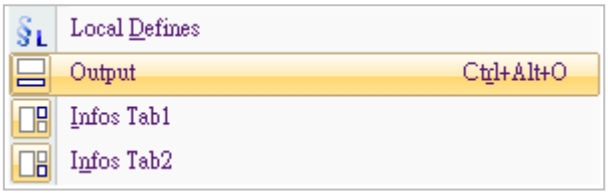
C. The Message Area: It used to show compiler messages and provides more diagnostic tools.

Tips:

1. To resize a window, click and drag the side or corner of the window to change its size.
2. Press the "F1" key to open the user manual (i.e., HTML Help).

Hind or Show the window

If you carelessly closed the Variables pane or the Message Area during the programming, you can click the menu bar "View" and select the following options to open this window again.



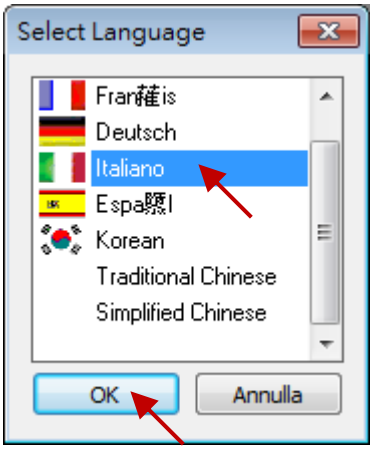
Output: It means the Message Area.

Infos Tab1: It means the Program Area – Variables pane (refer the Section 2.2.1).

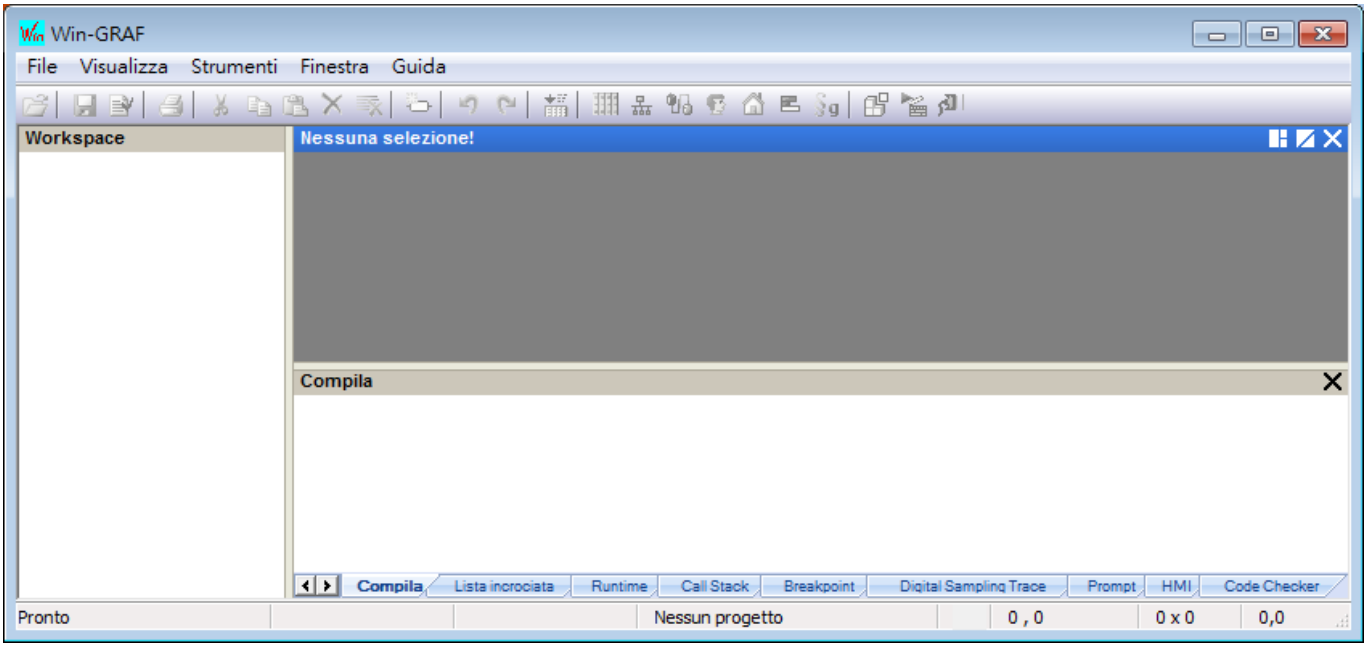
Infos Tab2: It means the Program Area - Function Blocks pane (refer the Section 2.2.1).

Switching the language

1. To switch the UI language of the Win-GRAF, click “Help” and “Language” on the toolbar and then choose a language in the “Select Language” dialog box, and click “OK”.



2. The UI language will be changed after restarting the Win-GRAF automatically.

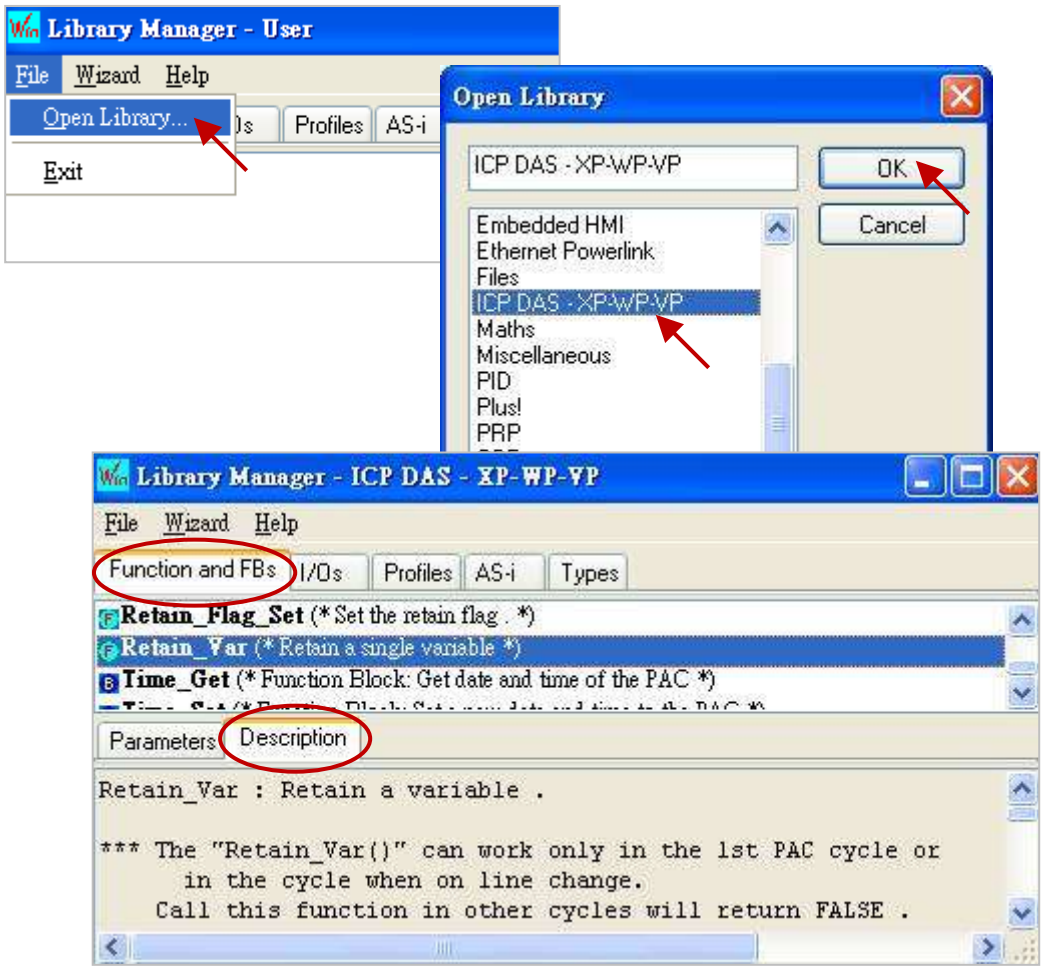


1.2.3 Win-GRAF Library Manager

Win-GRAF Workbench provides a Library Manager that can be used to look up all descriptions for Functions, Function Blocks and I/O Boards. The user can refer Section 11.1 to upgrade the Win-GRAF Lib.

1. To begin this, click the Start button and click "All Programs" > "Win-GRAF" > "Libraries" > “OEM”.

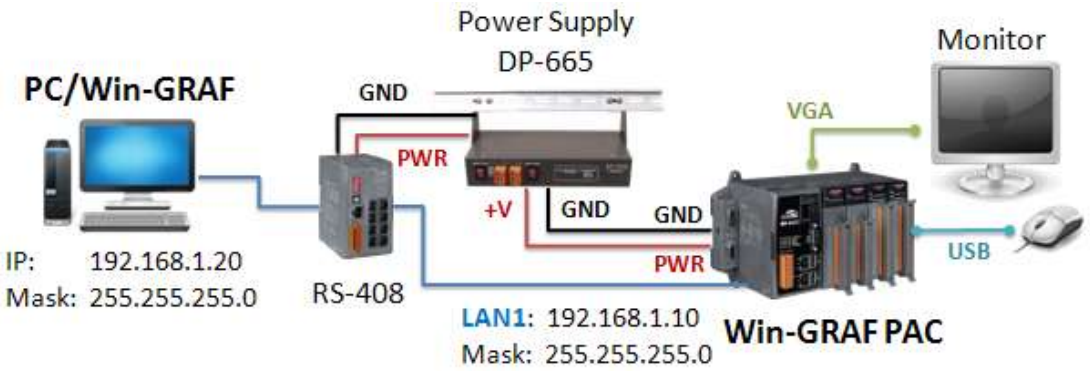
2. In the “Library Manager” window, click the menu bar “File” > “Open Library” and select “ICP DAS – XP-WP-VP” then click “OK”.
3. Select any title in the “Function and FBs” tab and click the “Description” to view the usage of this Function or Function Block; Select any title in the “I/Os” tab and click the “Description” to view the usage of this I/O Boards.



1.3 Setting the Win-GRAF PAC's IP Address

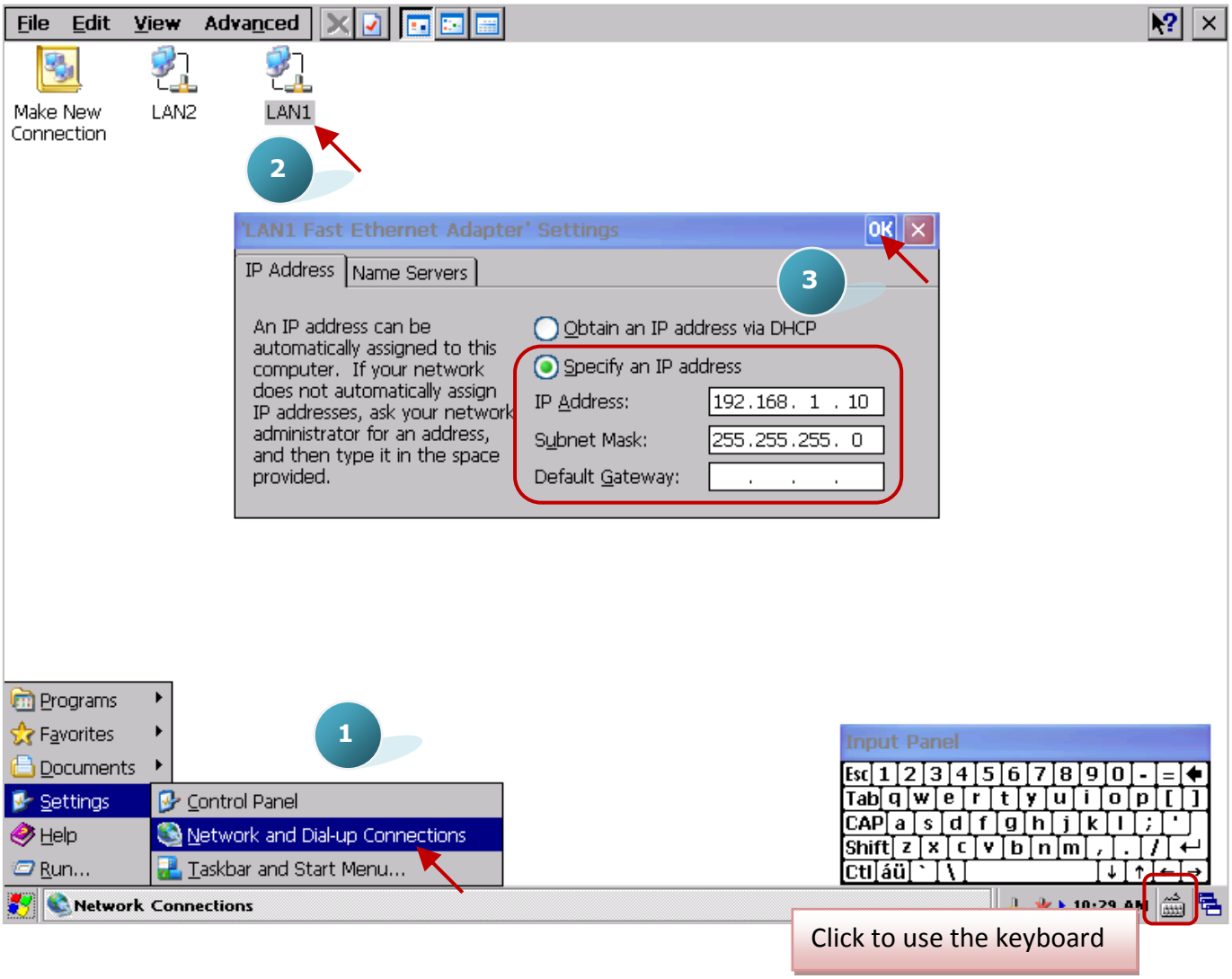
For connecting with the PAC, the Win-GRAF Workbench needs to know the PAC IP. The following will show you how to set up the PAC IP. Using the XPAC (XP-8xx8-CE6) and the WinPAC (WP-5xx8-CE7, WP-8x48, WP-8x28-CE7) as the example:

Hardware Wiring Diagram



PAC Side

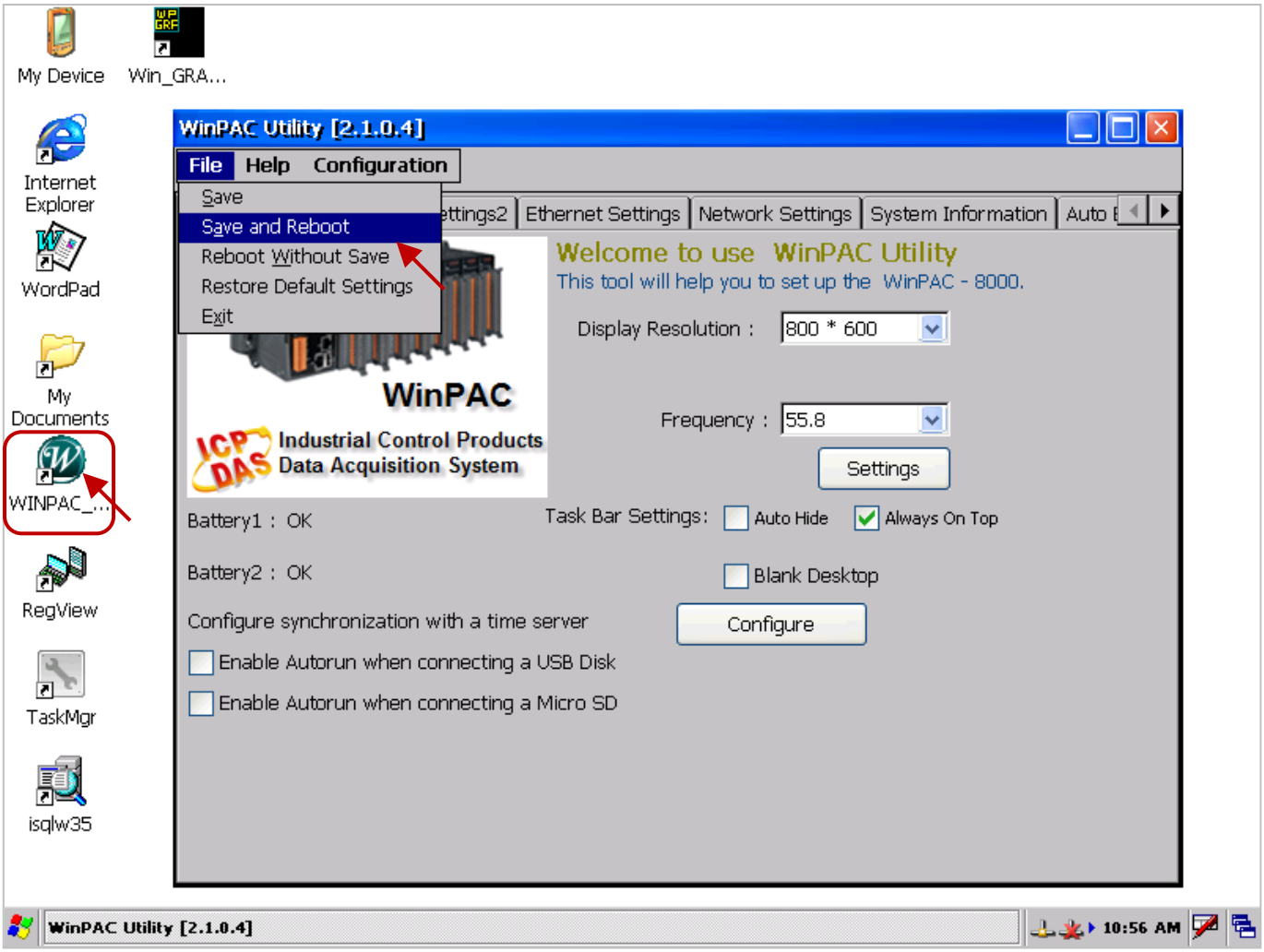
Using the USB mouse that connected to the PAC, and click "Start" > "Settings" > "Network and Dial-up Connections" on the lower left corner of the monitor, then double-click the "LAN1" (or LAN2), then fill in a proper IP address.



Open the “WinPAC_Utility.exe” (or “XPAC_Utility.exe”) on the desktop (or \System Disk\tools\).Then, click “File” > “Save and Reboot” to reboot the PAC.

Notice:

For connecting properly, the PC/Win-GRAF IP and the PAC IP must on the same network segment.
For example, set the PC’s IP to “192.168.1.20” (Mask: 255.255.255.0).



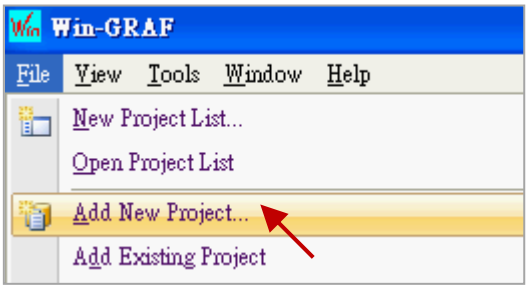
Chapter 2 A Simple Win-GRAF Program

2.1 Creating a New Win-GRAF Project

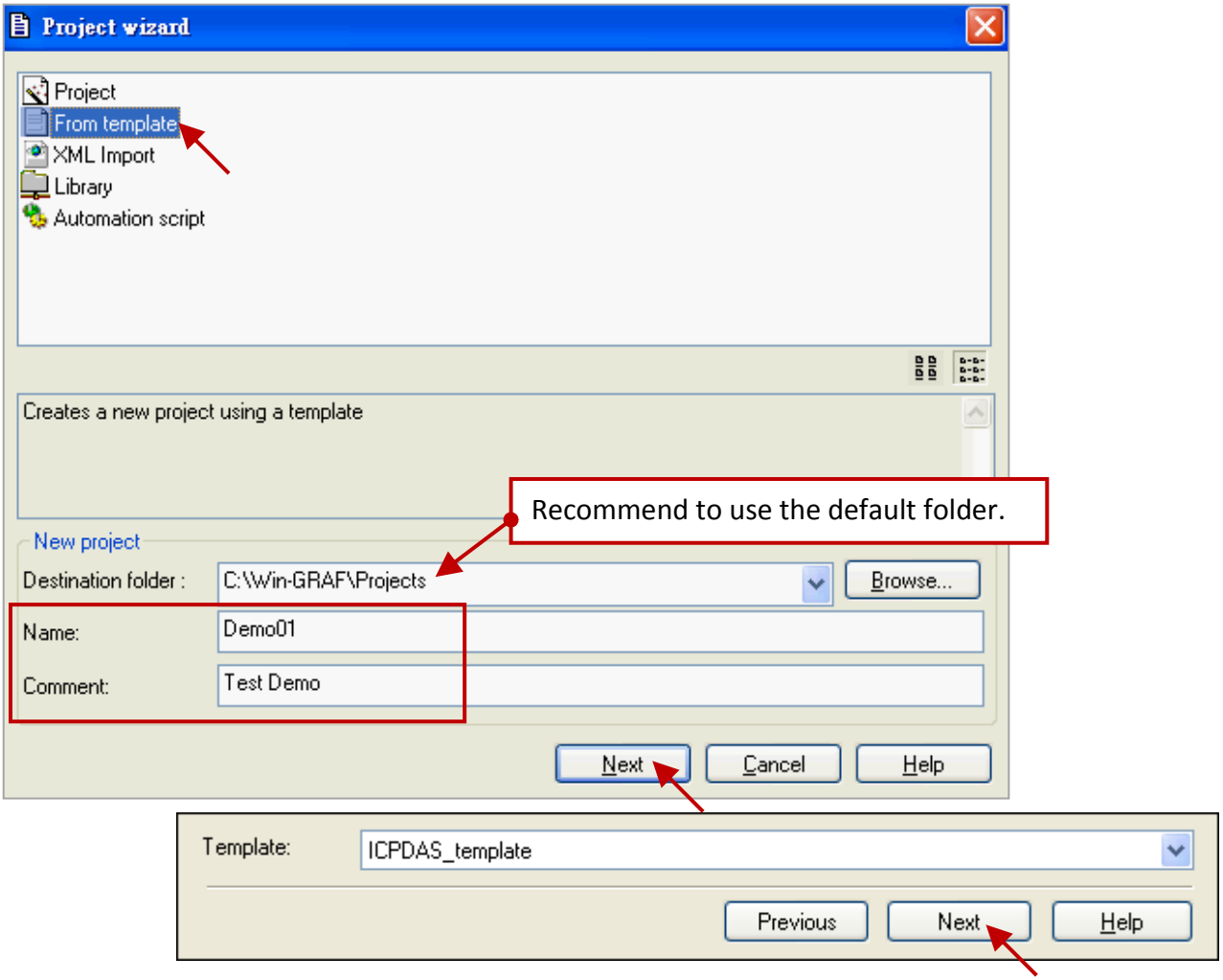
The following sections will introduce you to a simple template project that used to get/set (read/write) the Win-GRAF PAC's system time. Follow the steps below to complete this demo program.

2.1.1 Creating a Template Project (Demo01)

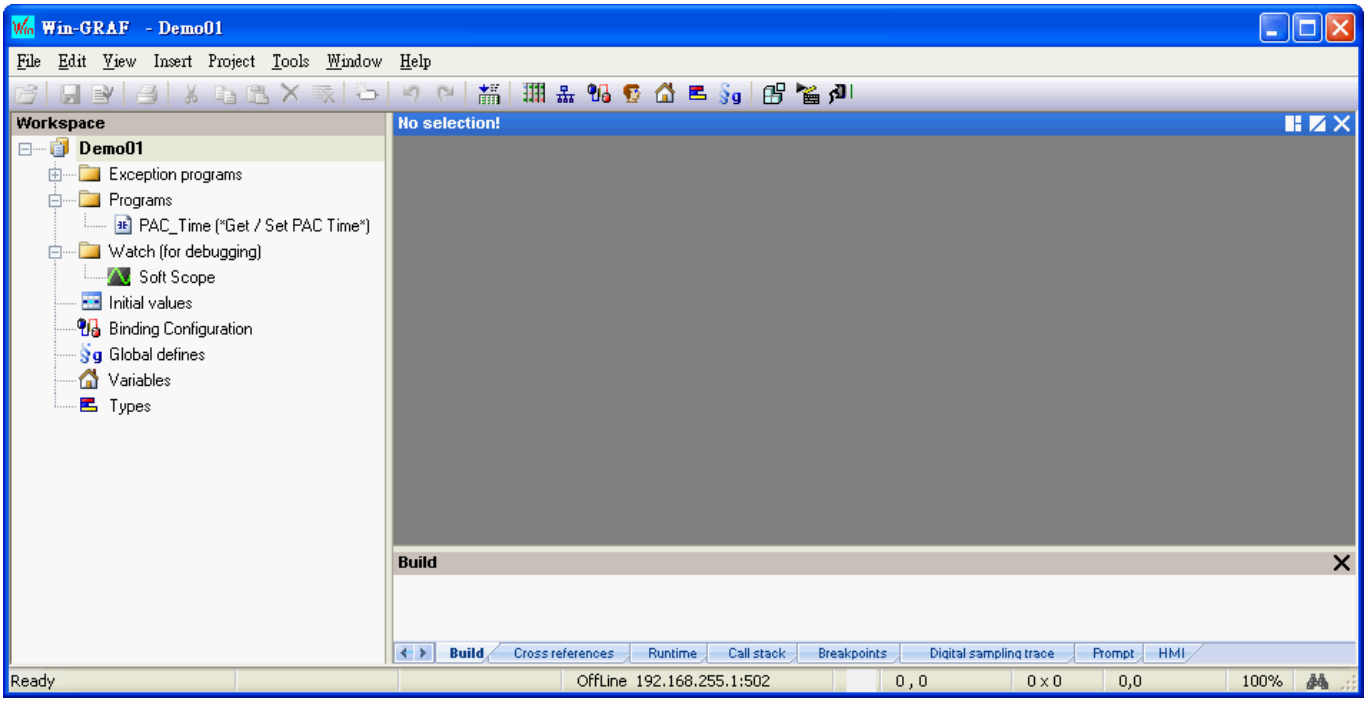
1. Run the Win-GRAF Workbench (refer the Section 1.2), and click "File / Add New Project..." from the menu bar.



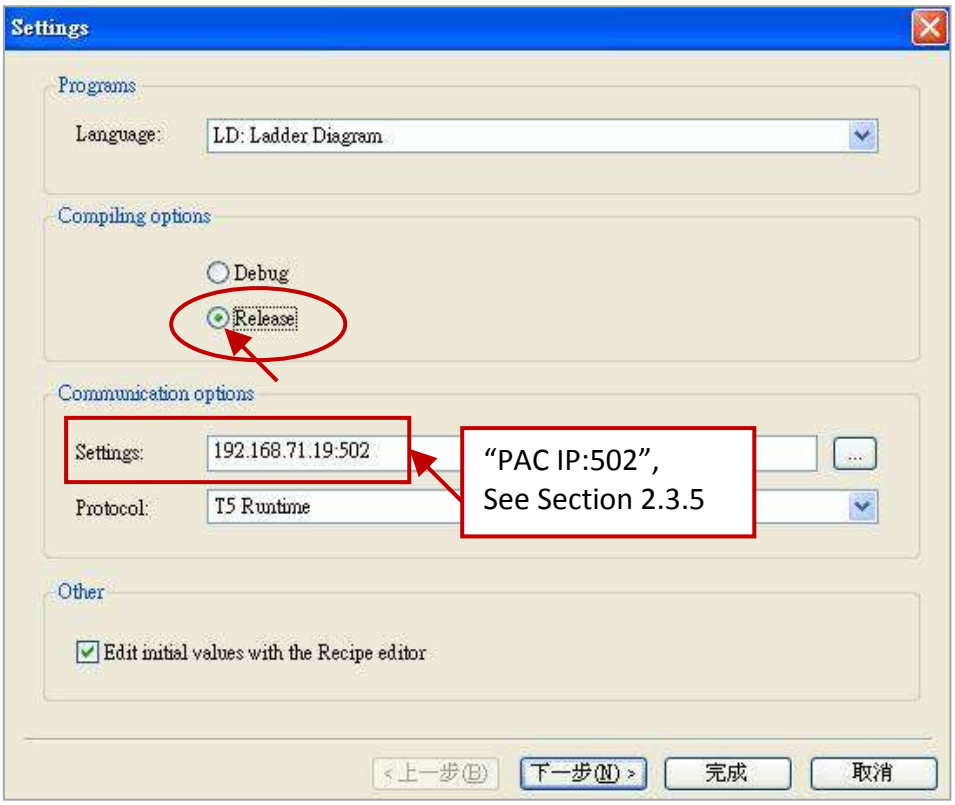
2. Click "From template" to create a project from a template, enter a project name (e.g., "Demo01") in the Name field and add a simple note in the Comment field, then click "Next". By default, it will show a "ICPDAS_template" option provided by Win-GRAF Workbench, just click "Next" to continue.



3. Now, you have created the “Demo01” template project.



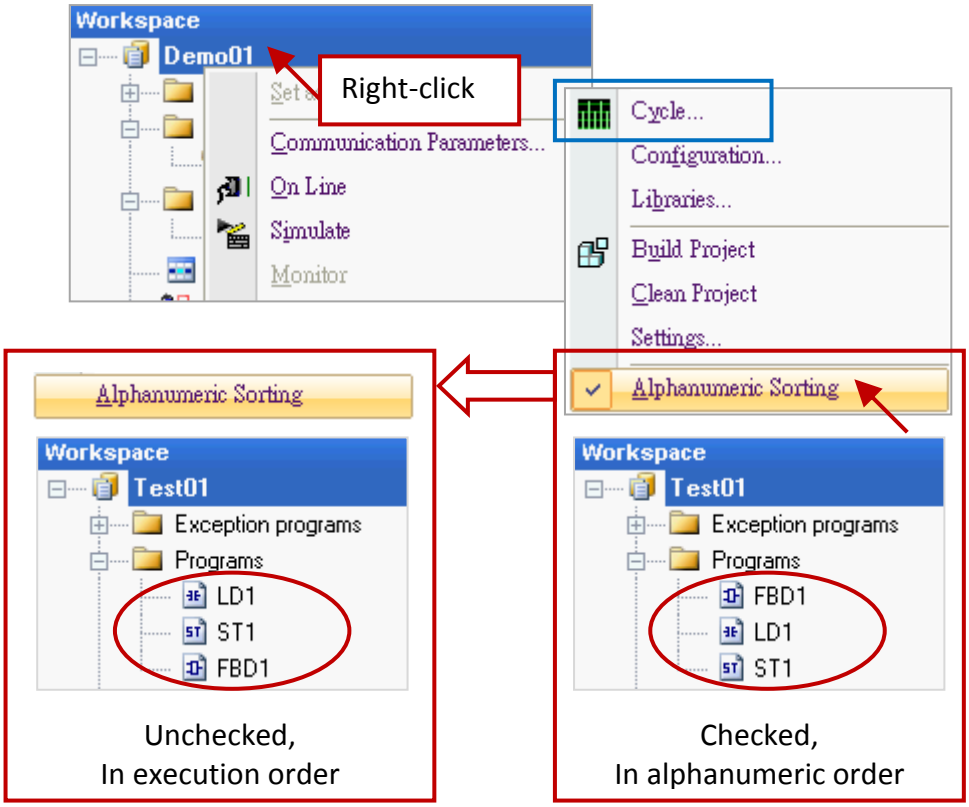
Note: In the demo01, we use a “From template” way to create this project. If you select "Project" in the step2, click the "Release" in the “Compiling options" setting. The others setting can be done in the following sections, just click "Next" and then "OK" button to end the settings.



2.1.2 Important Project Settings

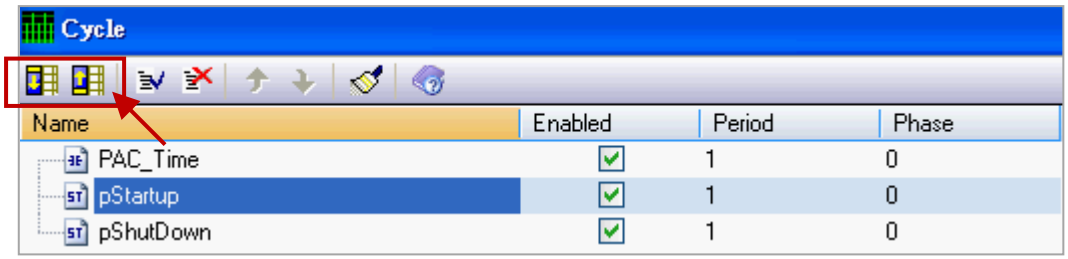
There are two important settings must be done after creating the project.

1. In the "Workspace", mouse right-click the project name (e.g., "Demo01") and then uncheck the "Alphanumeric Sorting" option (the last one). If unchecked, means the programs are in execution order ; If checked, means the programs are in alphanumeric order (e.g., FBD1, LD1, ST1).

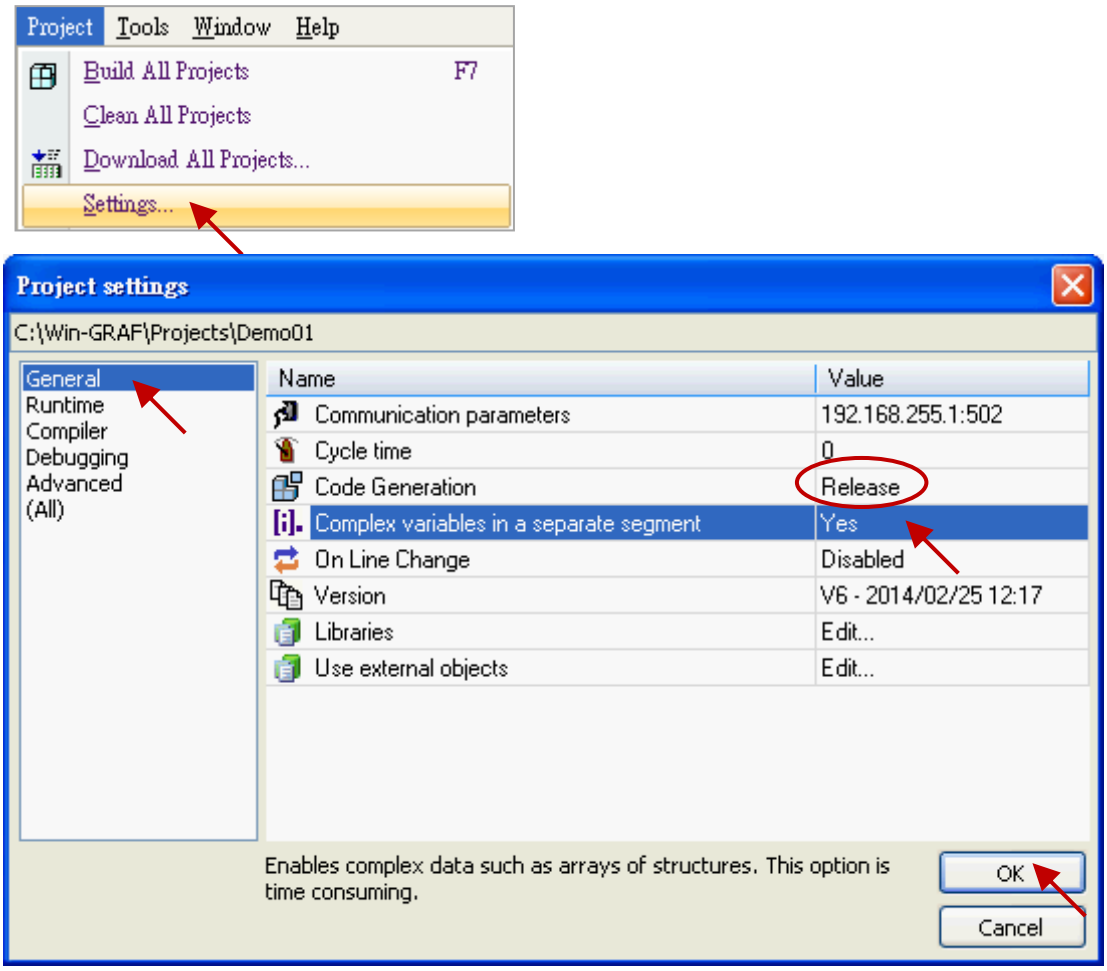


Change the execution order of programs:

Note: If you want to change the execution order of programs, mouse right-click the project name (e.g., "Demo01") and click "Cycle" (as the screenshot above) to open the settings window, then click the "Move Up" or "Move Down" button to change the order.



2. If using the "Project" way to create a new project (in this example, we use the “From template” way, refer the Section 2.1.1 - Step 2), click the"Project" > "Settings..." from the menu bar to open the "Project settings" window. Click the "General" option and set the "Complex variables in a separate segment" to "Yes" to allow the using of complex data structures, such as arrays. Finally, click "OK" to exit the window.

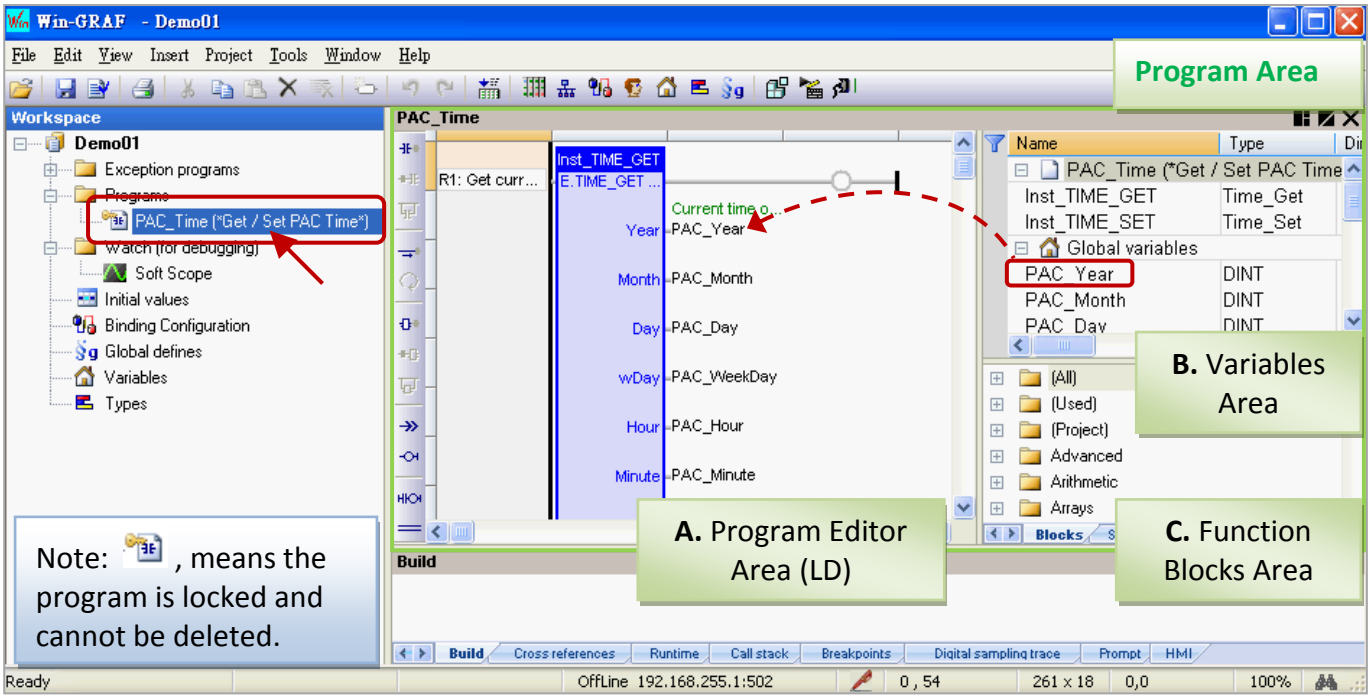


(**Note:** The "Code Generation" must be set to "Release".)

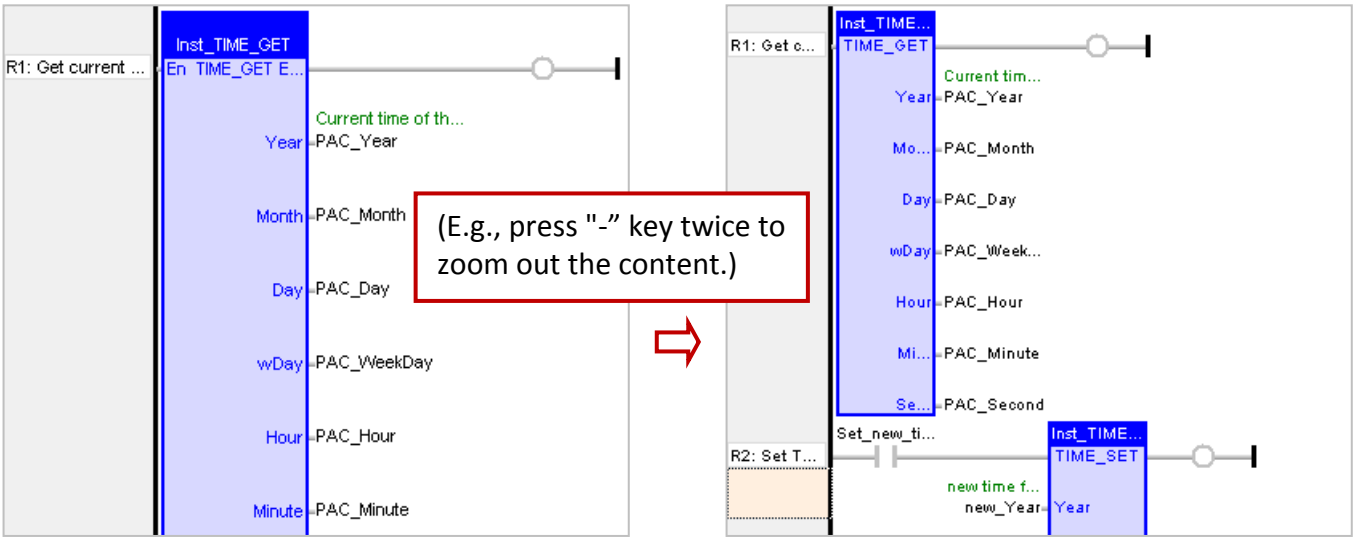
2.2 Introduction of the Project

2.2.1 Demo01 - LD Program

This program is used to read/write the Win-GRAF PAC's system time. In the Workspace, double-click the LD program name (i.e., "PAC_Time") to open all relevant windows. As the screenshot below, the Program Area has three main parts:



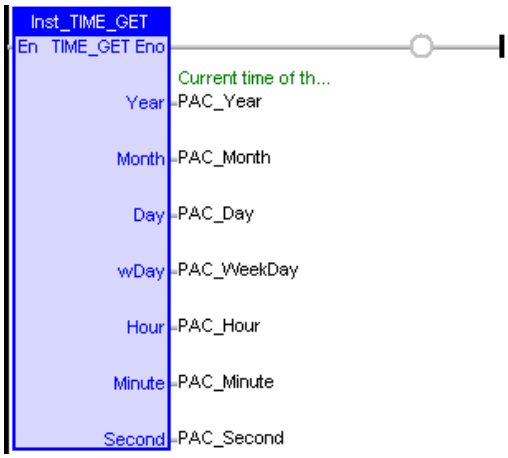
Tips: Mouse click the Program Editor Area, and press the "+" or "-" key to zoom in or zoom out the content.



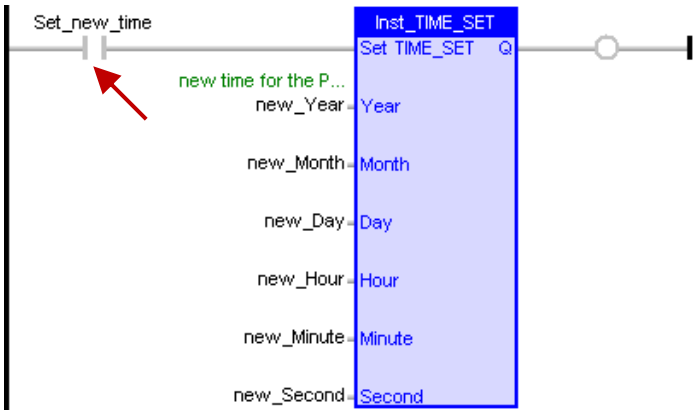
A. Program Editor Area (LD):

This area allows to edit or display this LD program, you can click the object button (on the left of Program Area) to add a program, and then drag-and-drop variables (in the Variables Area) onto the function block one-by-one.

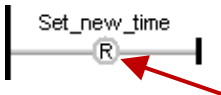
R1: Get current time of the PAC



R2: Set "Set_new_time" to "True" to set the new time



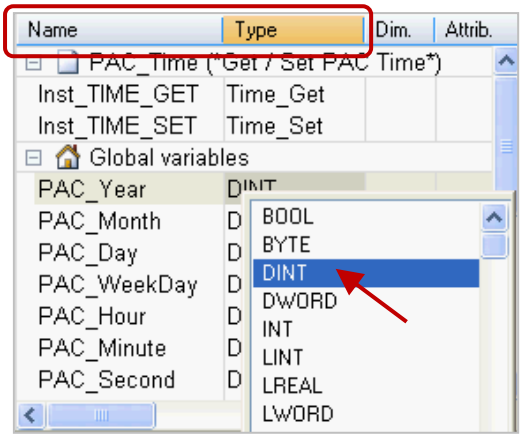
R3: Reset it to "False"



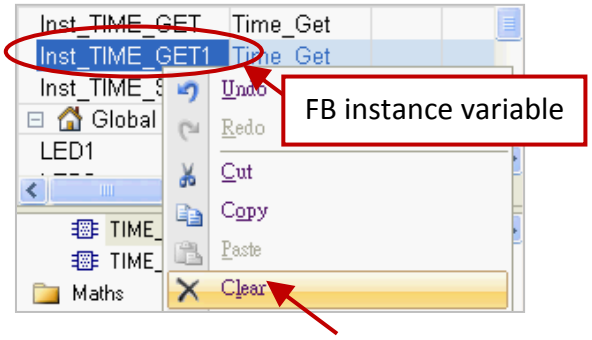
Note: You can press the SPACE bar to change its type (/, S, R), and press the F1 key for more details.

B. Variables Area:

This area shows the function blocks and variables that used in this program. Mouse double-click the "Name" or "Type" item to modify its name or data type, then click "Enter" to complete the setting. (Refer the Section 2.3.1 for the details about variable declaration.)

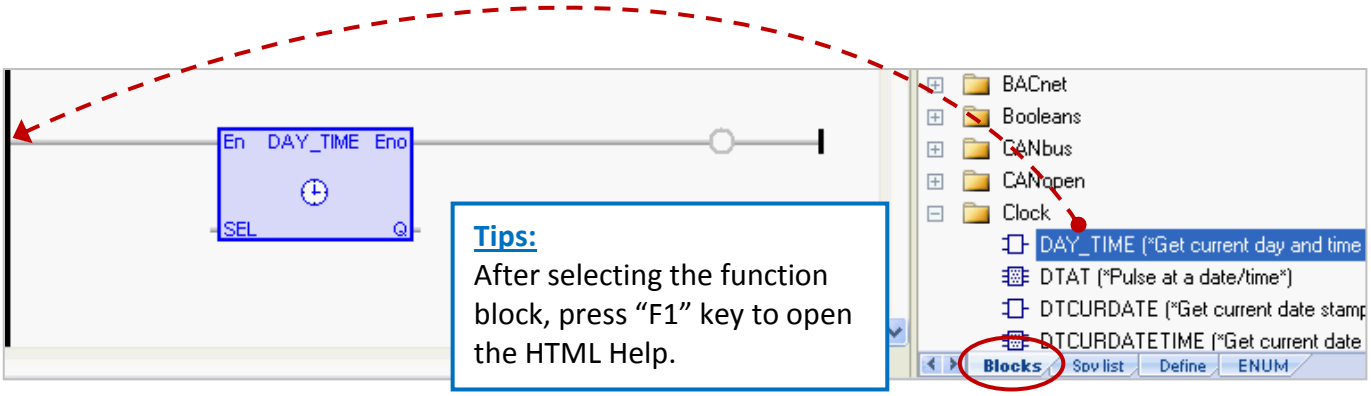


Note: For the function block can work correctly, the "Inst_xxx.." FB instance variable will be automatically added when using one function block. For the safety reasons, this FB instance variable will not be automatically deleted even if the function block has been removed in the editor area. So, users can right-click the unwanted variable and select "Clear" to manually remove it.



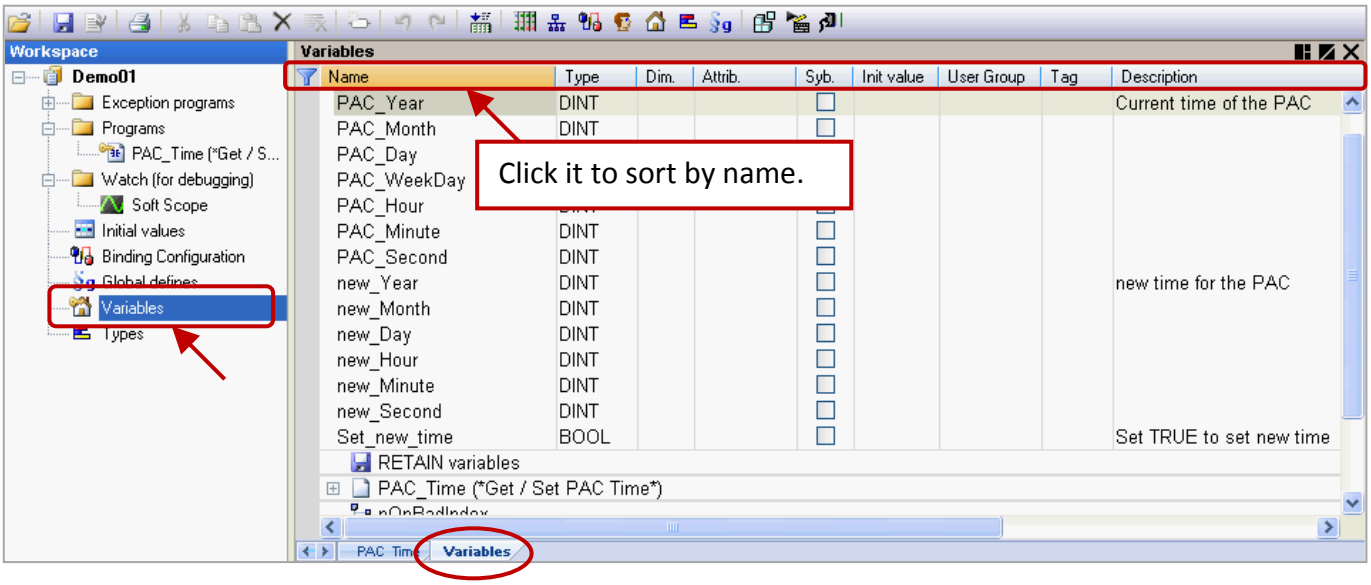
C. Function Blocks Area:

In the "Blocks" tab, it provides many types of the function block for users to drag and drop them to the editor area.

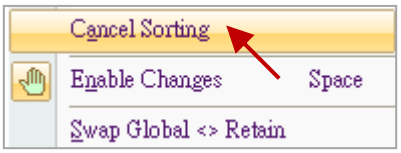


2.2.2 Demo01 - Variables

In the workspace, mouse double-click the “Variables” item to open the Variables window. The following screenshot shows all the needed and defined variables in this “Demo01” project.



Tips: In the Variables window, users can click any title field (e.g., “Name”) for sorting.
If you want to go back to the original sort order, mouse right-click anywhere and select the “Cancel Sorting” option.



Field description: (Press the “F1” key to look up the details)
Mouse double-click any field item to set or modify the data.

Name:	A valid variable name starts with a letter (e.g., "A to Z" or "a to z") followed by any number of letters, numbers (e.g., "0 to 9"), or an underscore (i.e., “_”).
Type:	Data type. (Refer the Appendix A for the value range)
Dim.:	To specify the range of an array. (E.g., enter “10”, means the use of the Counter [**0**] to [**9**]).
Attrib.:	Double-click this field item to set it to “Read Olny” that means users can only read this variable but cannot modify it.
Syb.:	If checked, the variable name will also be downloaded into the PAC.
Init value:	To set the initial value of the variable.
User Group:	All the variables can be divided into some groups (e.g., “Group1”, “Group2”) and it is convenient for users to look up or search these variables.
Tag:	To enter a nickname for the variable.
Description:	To enter a simple note for the variable.

2.3 Give it a Try

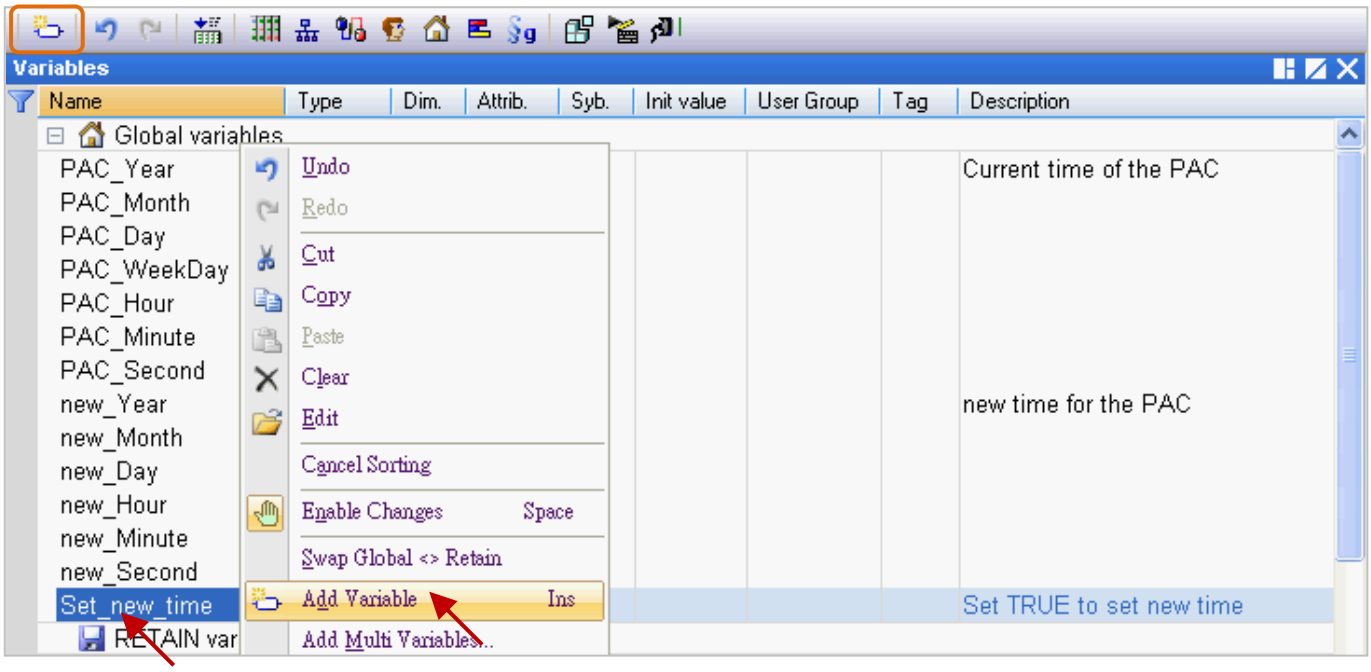
As mentioned before, we have described the LD program (Section 2.2.1) and variables (Section 2.2.2) in the "Demo01" project. The following sections will show you how to declare variables and add an LD program with the blinking function in this project.

Note: All the Win-GRAF PAC does not support the "ULINT" and "LWORD" data type.

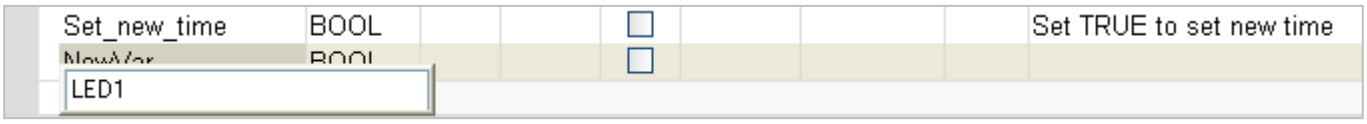
2.3.1 Declaring the Win-GRAF Project Variables

First, we will declare two boolean variables (i.e., "LED1" and "LED2") that used in the program.

1. In the "Variables" window, mouse right-click any item in the "Name" field and select "Add Variable" (or press the "Ins" key or click the  tool button) to add a variable.



2. Double-click the new "NewVar" item and change its name to "LED1", then click "Enter" to finish the setting. In this case, the data type is "BOOL".



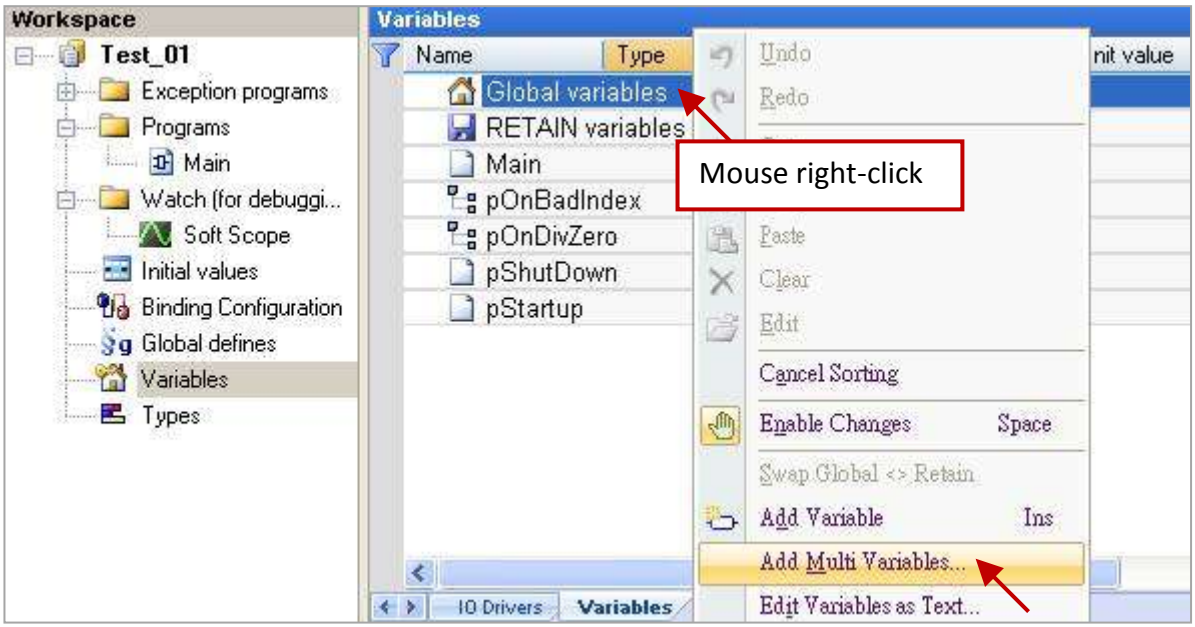
Note: The settings will be done only after clicking the "Enter" key.

3. Follow the previous steps to add the "LED2" boolean variable.

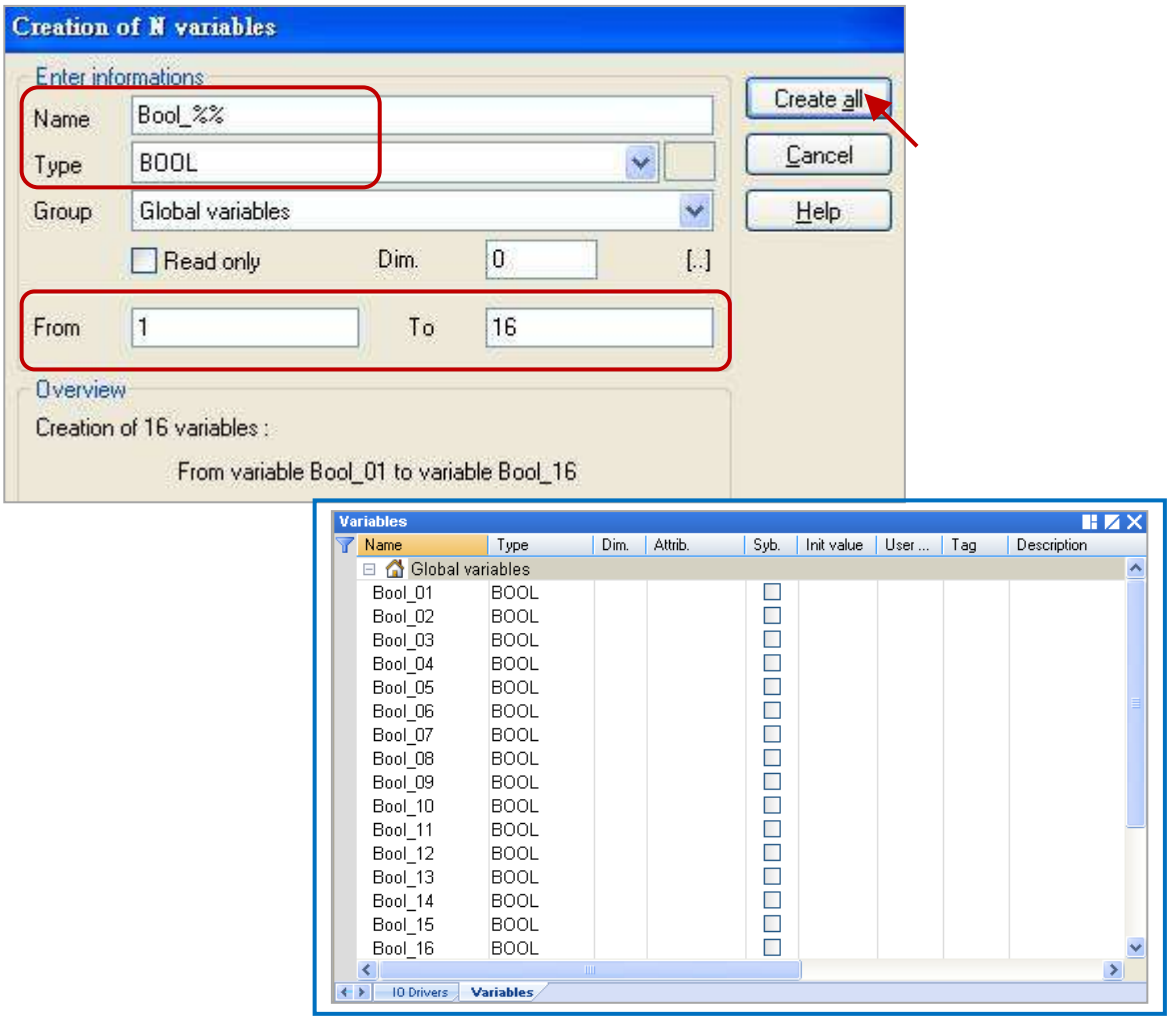
Tips: To set up multiple ordinal variables, enter the name "LED" (as the step2) and then press "Ctrl+C" and "Ctrl+V" twice to create "LED**1**" and "LED**2**" (auto sequential numbering), finally, delete the first variable (i.e., "LED").

Tip #2:

1. If you need to add multiple variables (e.g., “Boo_01 to Boo_16”), simply right-click the “Global variables” and select the “Add Multi Variables”.



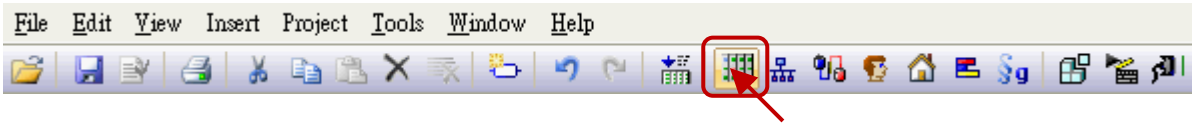
2. Follow the settings like the figure below (Name: “Bool_%%”; Type: “BOOL”; From: 1; To: 16) to create Boolean variables (i.e., “Bool_01” to “Boo_16”) and then click “Create all” button to complete the settings.



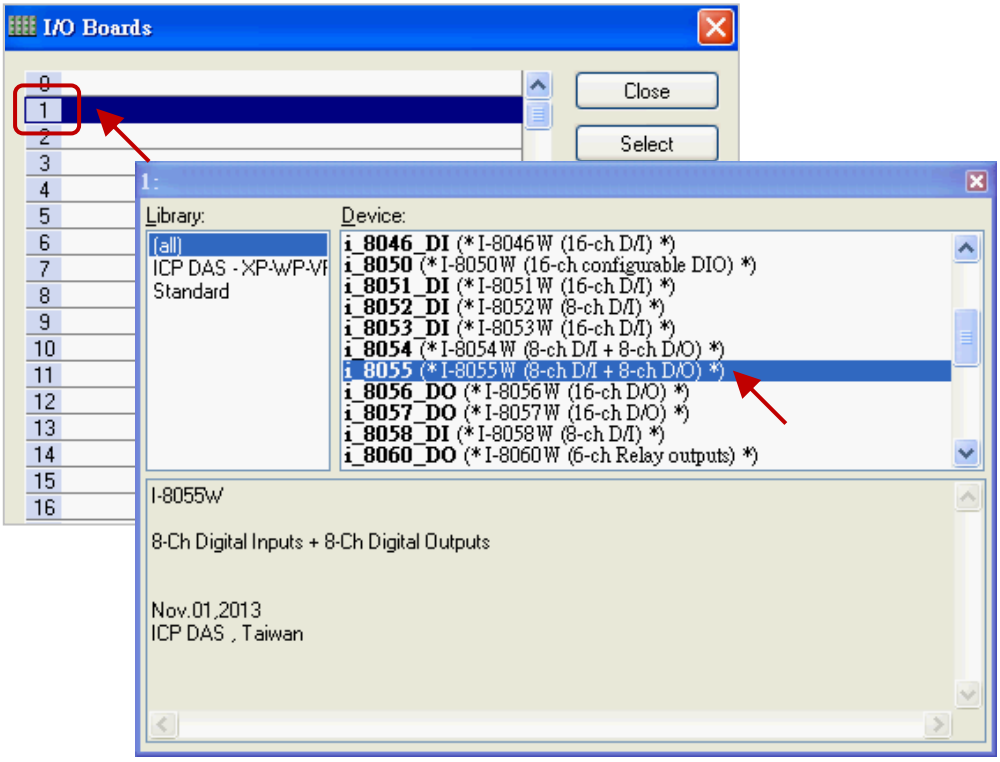
2.3.2 Declaring the I/O Variables

In this example, the I-8055W module that used to show the blinking feature must be plugged in the PAC's slot1. So, we need to add an I/O link to correspond to the real I/O module.

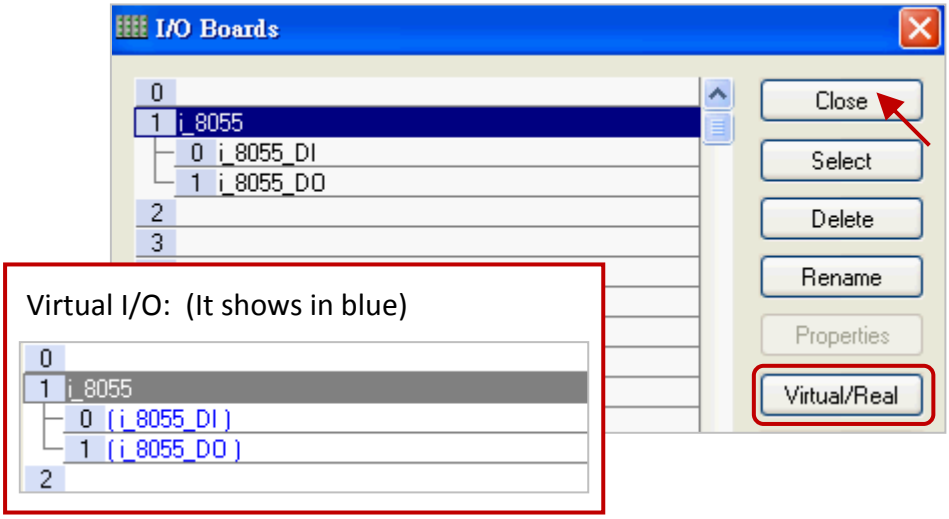
1. Click the "Open I/Os" tool button to add an I/O link.



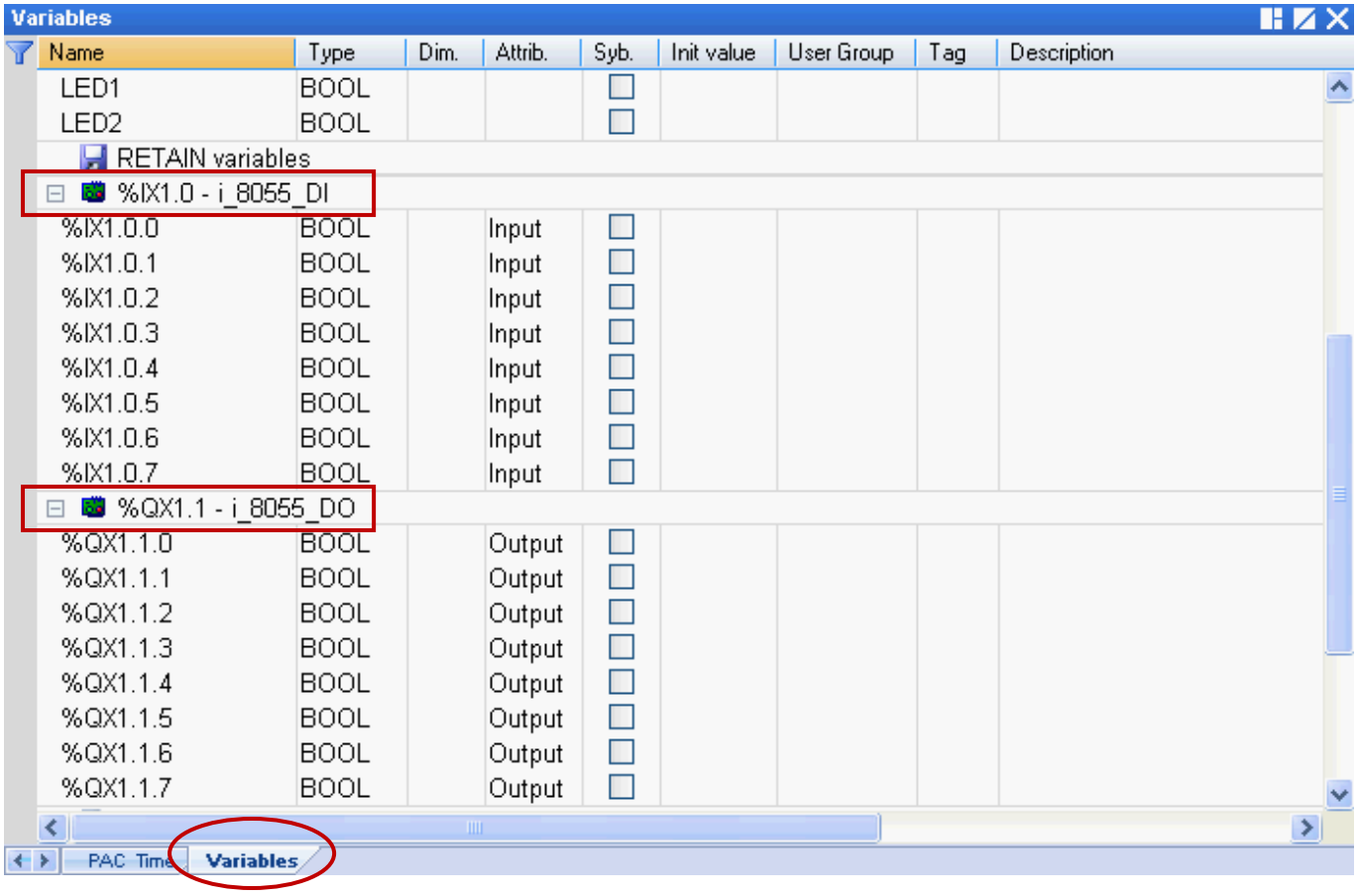
2. Mouse double-click on "Slot 1" and then double-click the "i_8055" to select this I/O board.



3. Click the "Close" button to exit the "I/O Boards" window.
   Note: Click the "Virtual/Real" button to change to the Virtual I/O (for testing) or the Real I/O. (The Real I/O is used in this example).



After linking the “i_8055” I/O board, it will automatically add 8 input & output variables in the “Variables” window.



%IX1.0 – i_8055_DI

“I” means **“Input”**

“X” means **“Boolean”**

“1” means **“Slot 1”**

%QX1.1 – i_8055_DO

“Q” means **“Output”**

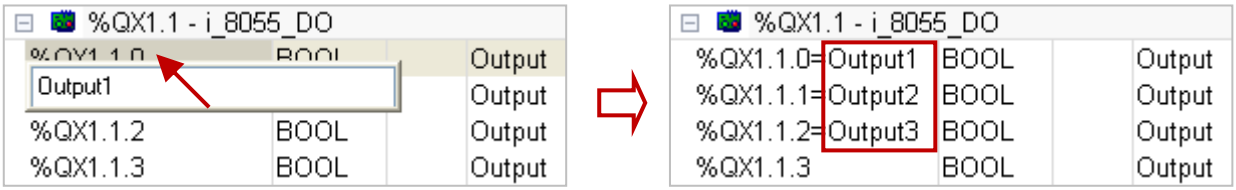
“X” means **“Boolean”**

“1” means **“Slot 1”**

%ID or %QD

“D” means **“Integer/Real”**

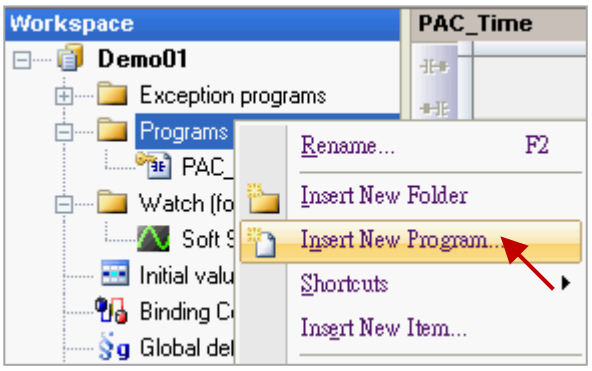
There are three output variables are used in this example, and you can modify the name for easy use. Mouse double-click the item and fill in a name, then press “Enter” key to finish the setting.



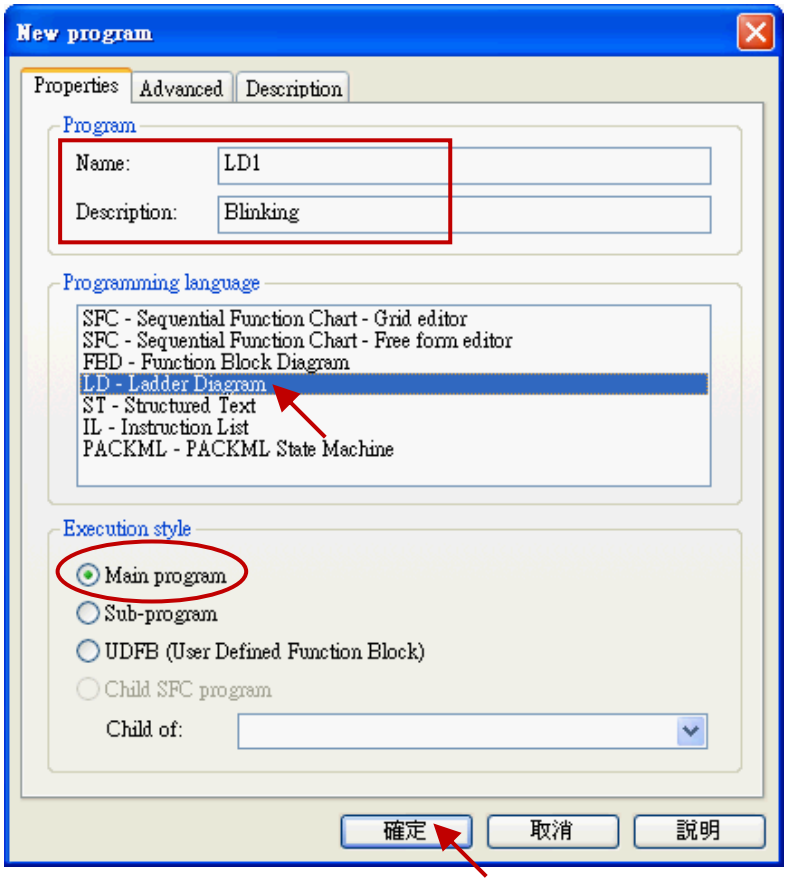
2.3.3 Creating an LD Program

In the “Demo01” project, we want to create a “LD1” program to show the blinking. To begin, follow these steps:

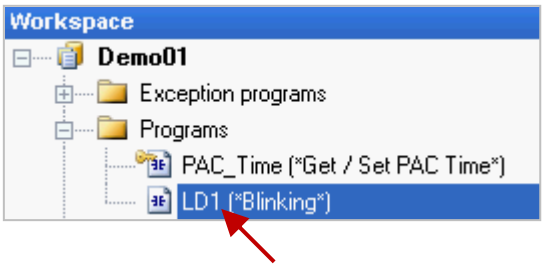
1. In the workspace, mouse right-click the “Programs” folder and select “Insert New Program…”.



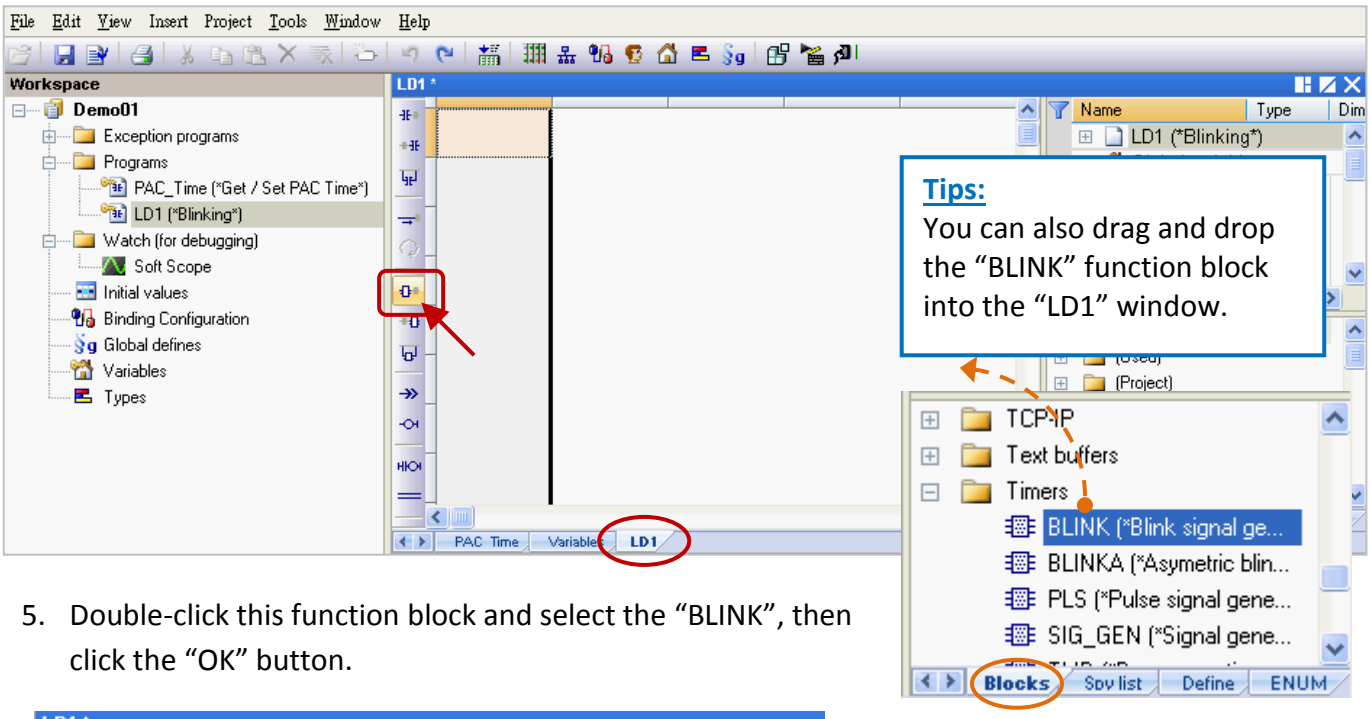
2. Fill in a program name in the “Name” field and enter a simple note in the “Description” field, and then select the “LD – Ladder Diagram” as the programming language and click the “OK” button.



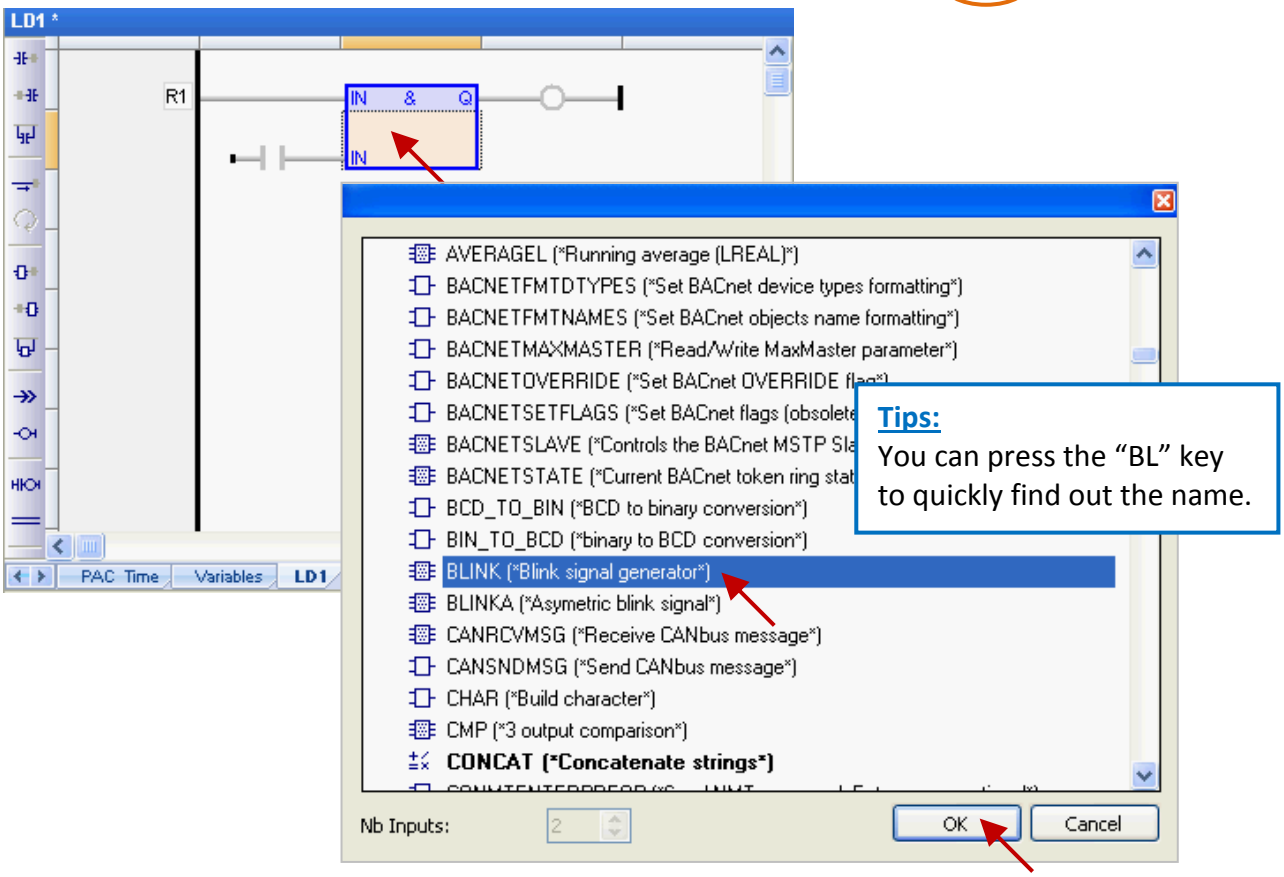
3. Double-click the “LD1” program to open the editor window.

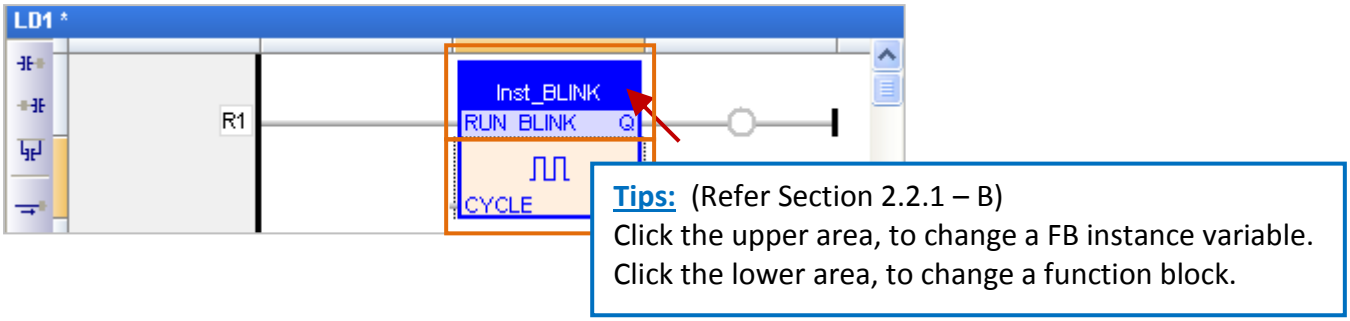


4. Click the "Insert FB.." button on the left of the "LD1" window to add a function block.



5. Double-click this function block and select the "BLINK", then click the "OK" button.

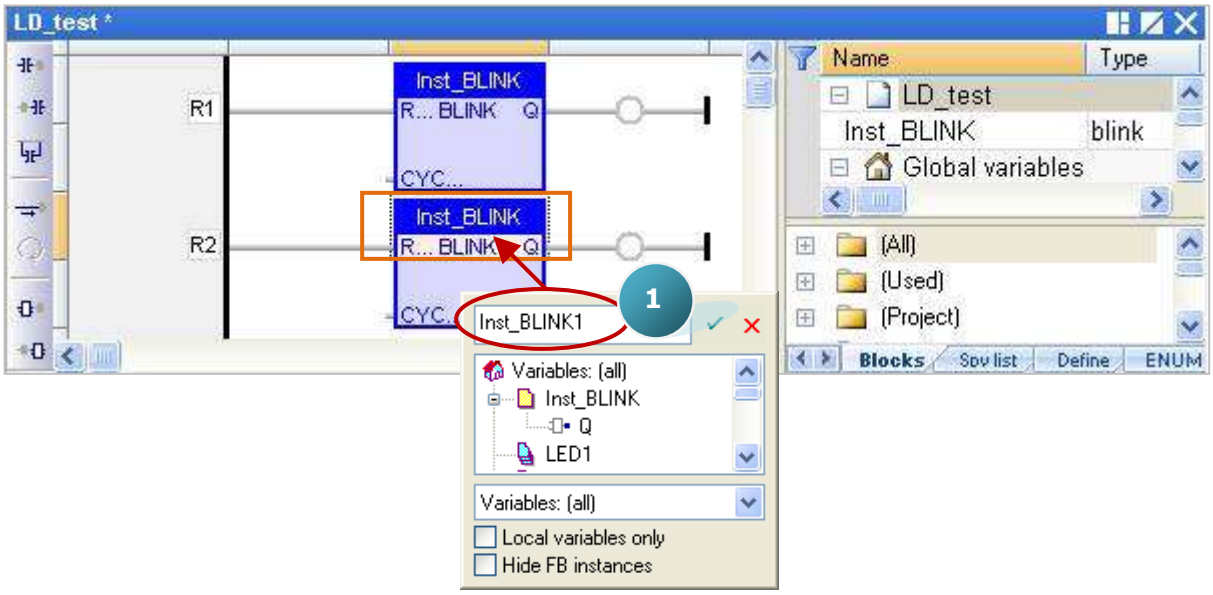




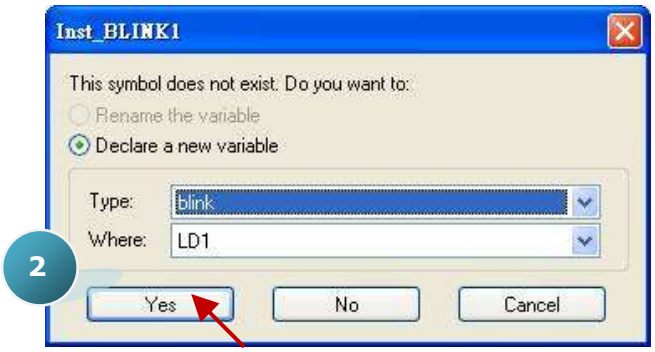
Important Notice:

When programming, users may copy & paste an existing function block to create a new one. But, this way will cause a function exception due to the same function instances. Therefore, users must create a new name for the function instance.

1. Mouse double-click the function block and enter a new name (e.g., "Inst_BLINK1"), then click the ✓ button to complete the setting.

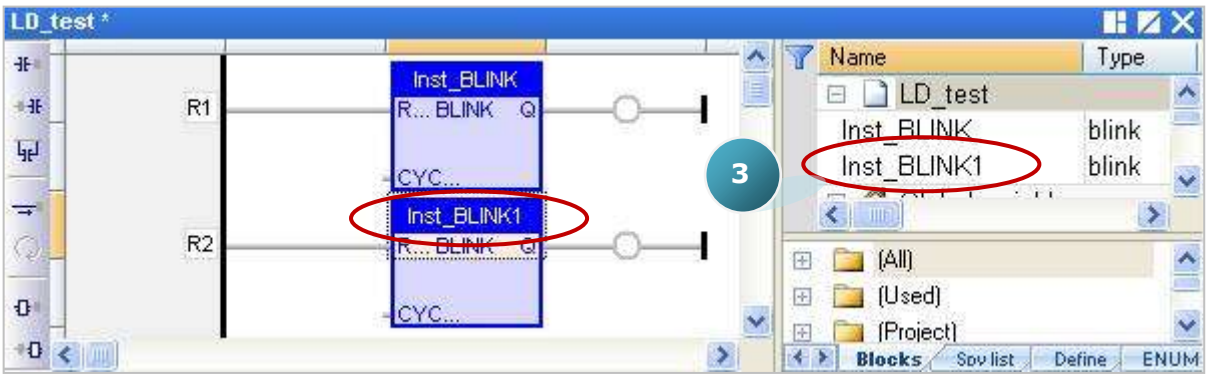


2. In the "Inst_BLINK1" window, click "Yes" to create this function instance.

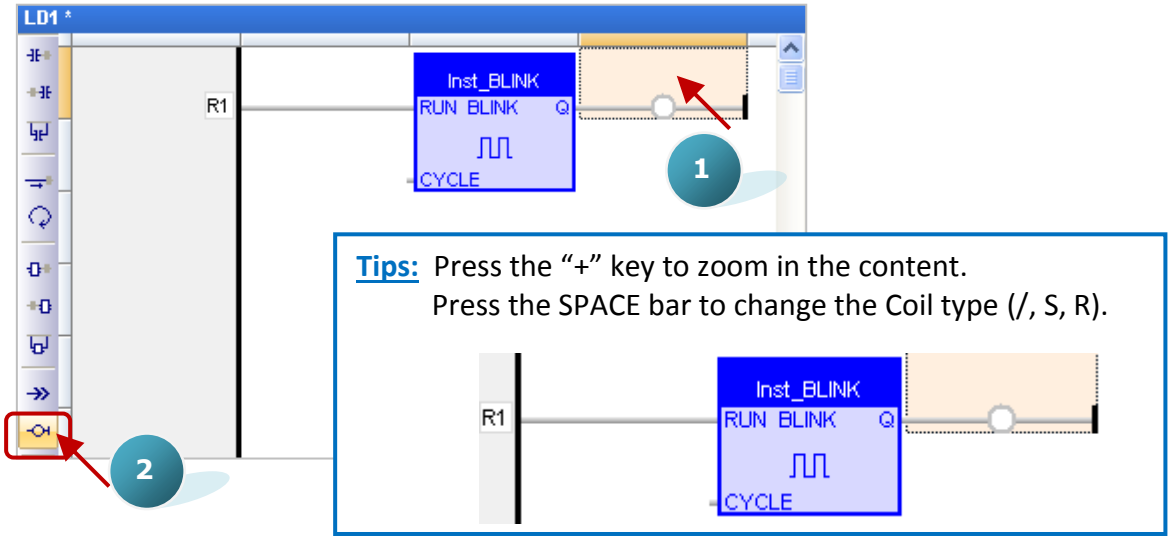


Tips:
You can also use the same way to double-click on the right-side of the "Coil" to create or assign a variable. (See Step7)

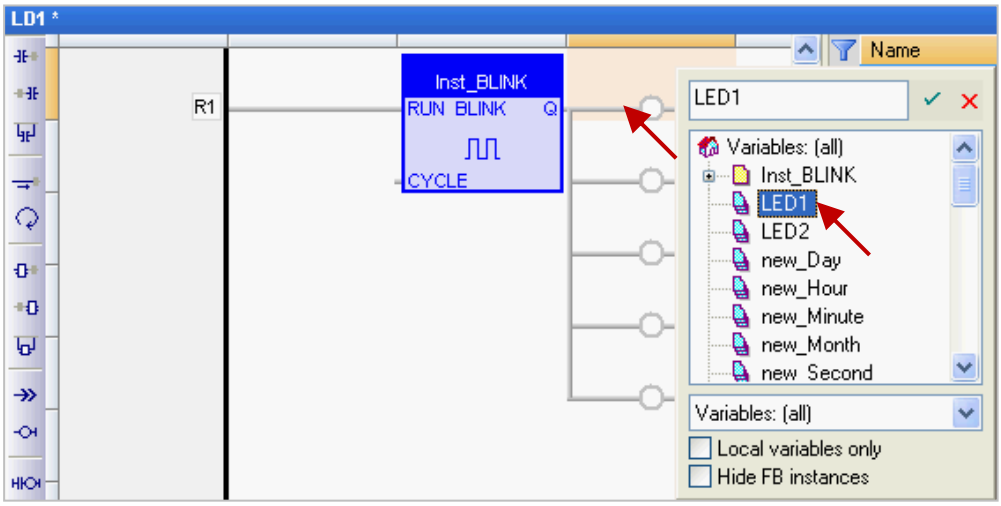
3. Now, there is an "Inst_BLINK1" function instance added in the Variables Area.



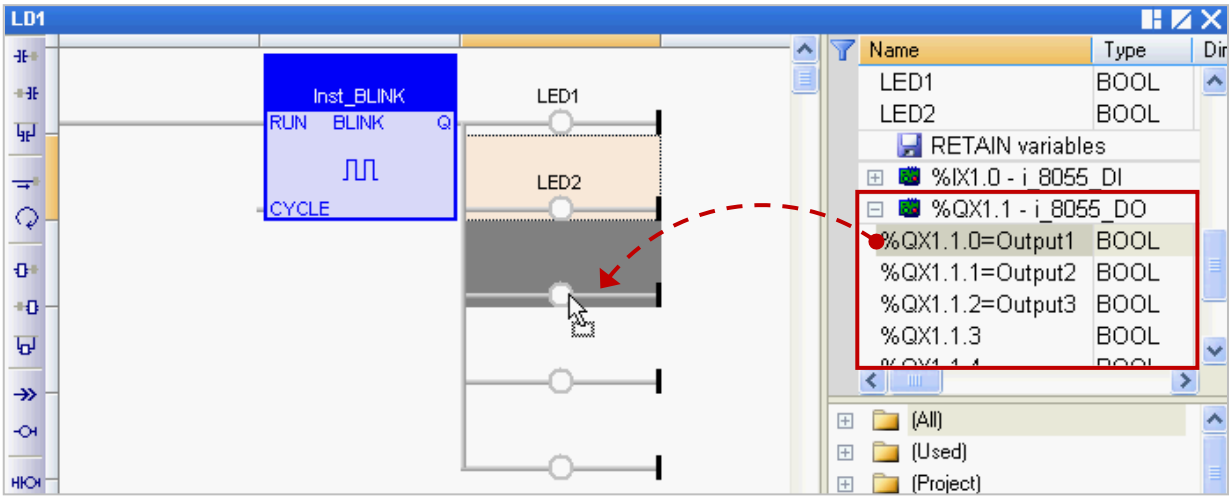
6. Click the “Coil” on the right of the “BLINK” function block, and continuously click the “Insert Coil” button to add four “Coil”.



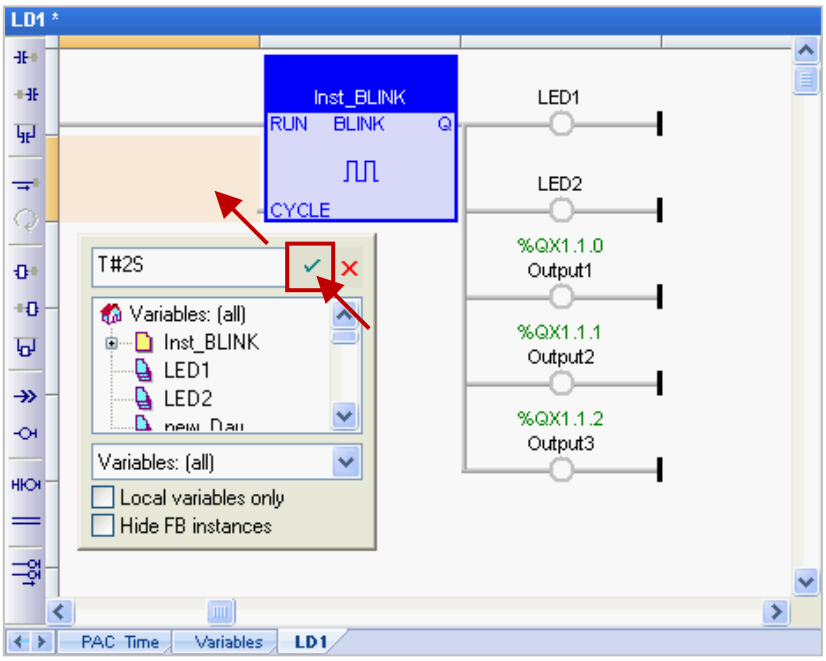
7. Mouse double-click the first “Coil” and double-click “LED1” to assign it. Follow the same way to assign the “LED2” variable to the second “Coil”.



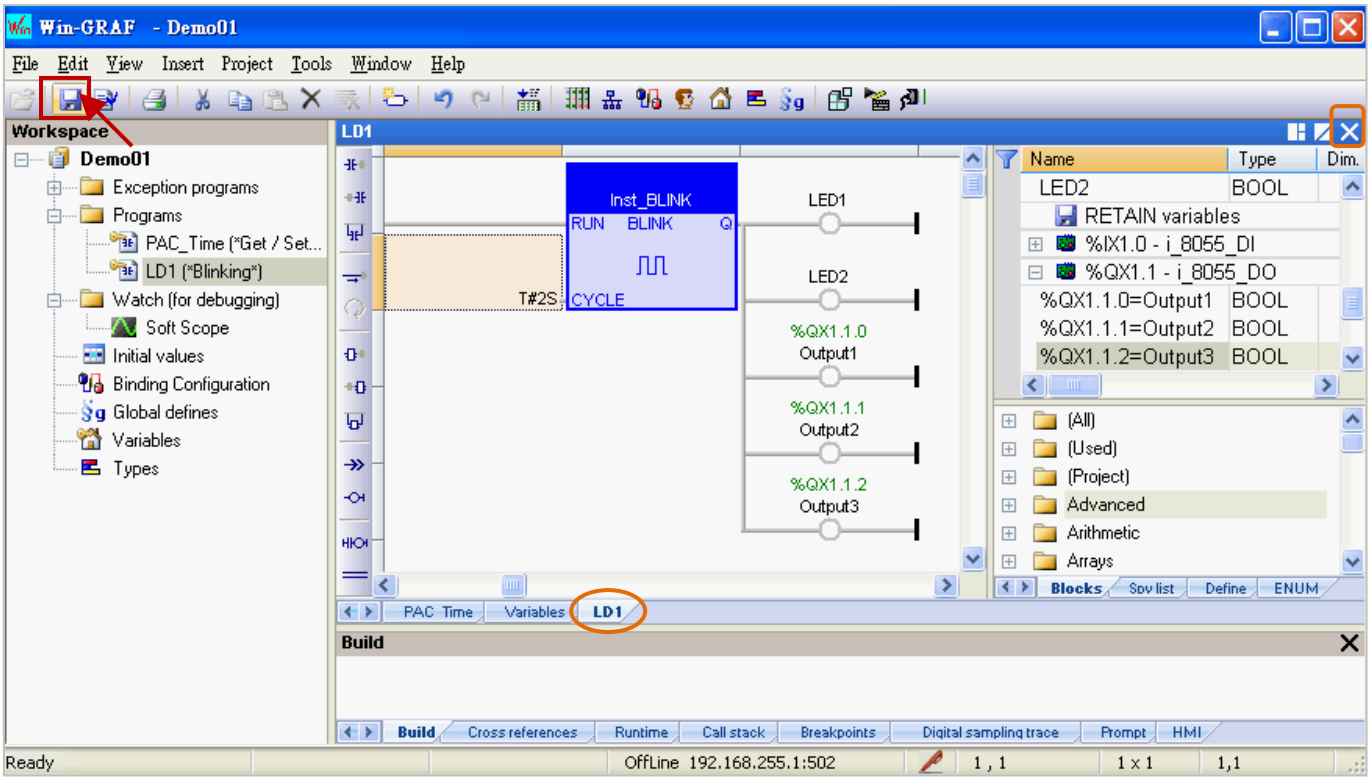
8. As the screenshot below, mouse drag-and-drop the “Output1”, “Output2” and “Output3” variables to the 3th, 4th and 5th “Coil”.



9. Mouse double-click on the left of the “CYCLE” and enter “T#2S” (to blink every two seconds), and then click ✓ to finish the setting.



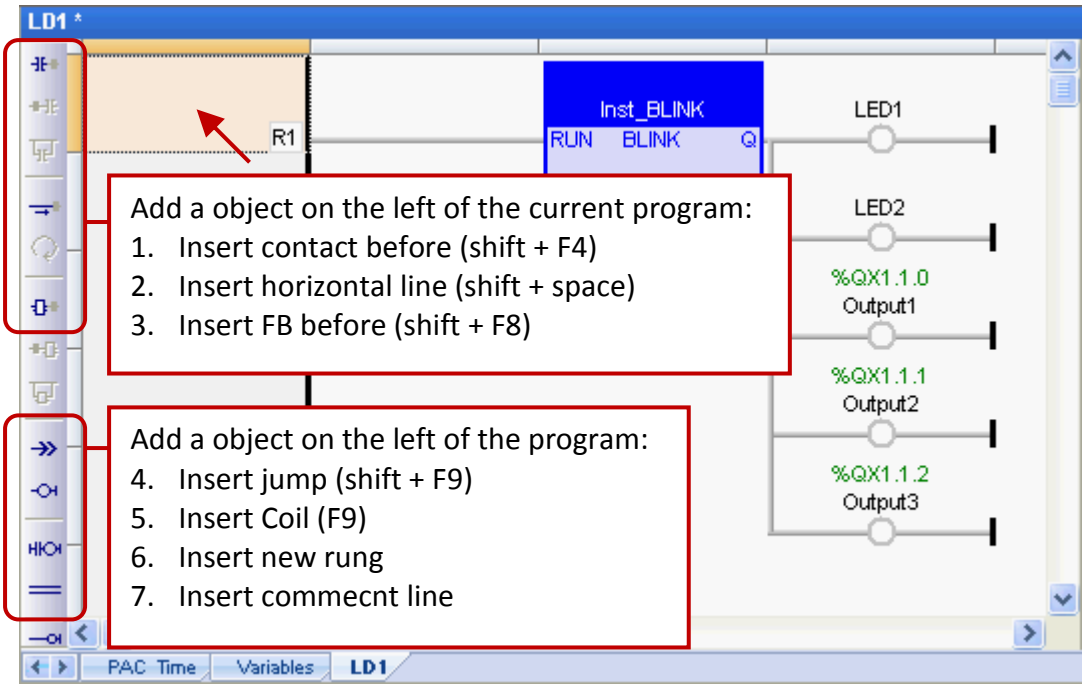
10. Finally, click the “Save” button to save the “LD1” program.



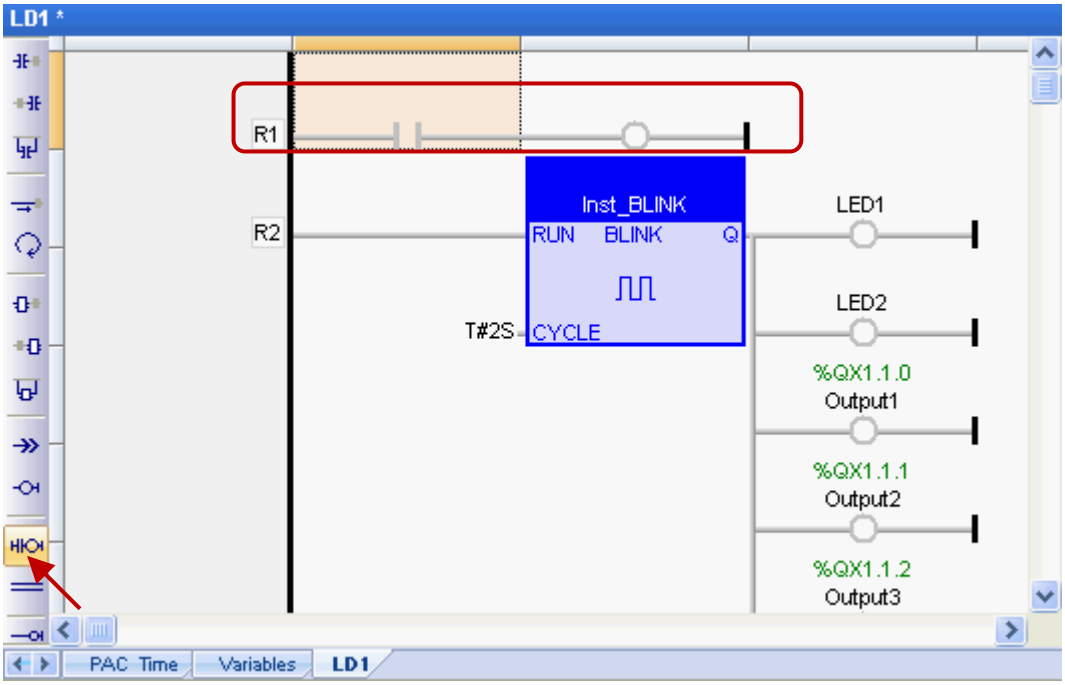
Note: “ ” means this program is opened (locked, cannot be deleted). Click the “X” in upper-right corner of window to close this program (un-locked, “ ”).

If you want to add a program to the first line after completing the programming, follow these steps:

1. Click the upper-left corner of the Program Editor Area, and you will see the selectable items on the Object bar. Select an object (4 to 7, as the screenshot below) to add a program to the first line.
   Note: You can also select an object (1 to 3) to add it on the left of the current program.



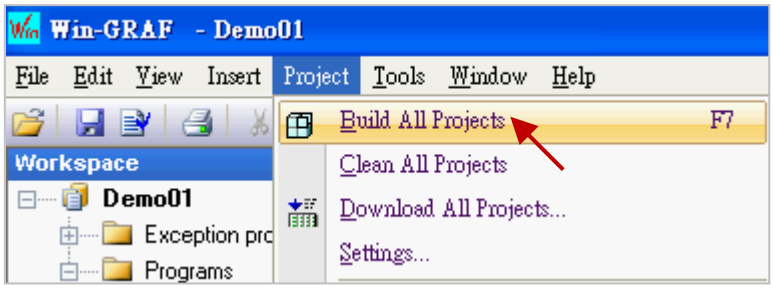
2. In this case, click “Insert new rung” to add it to the first line.



2.3.4 Compiling the Program

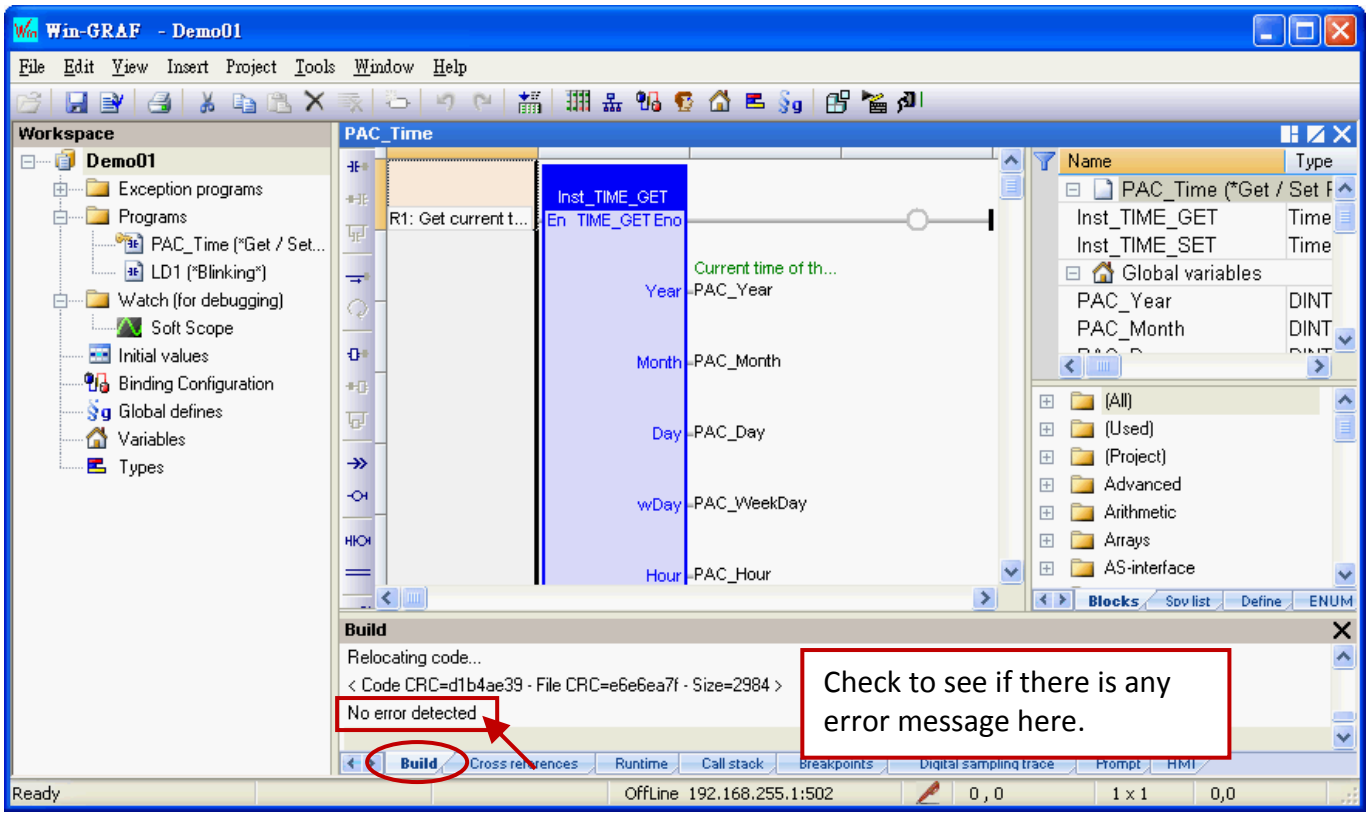
In the previous section, we have added and saved the LD program. For the Win-GRAF project can function properly on the PAC, we need to compile the programs. To begin, follow these steps:

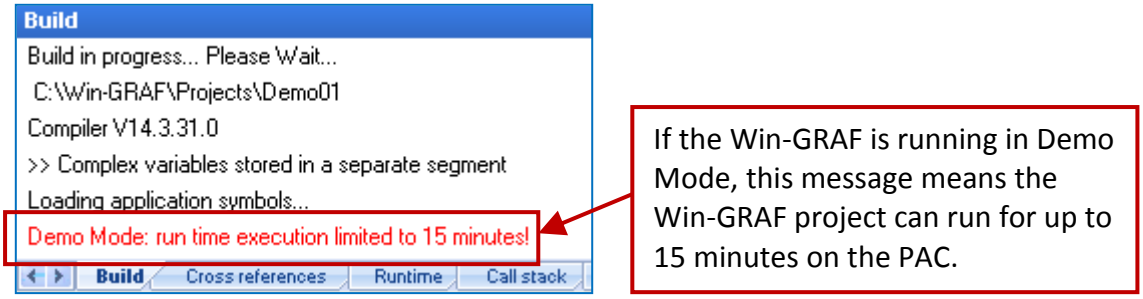
1. On the menu bar, click “Project > Build All Projects” to compile all programs.



2. If a “No error detected” message is appear that means the project was successfully compiled.

Note: **If you modify and save the program after compiling it,** click the “Clean All Projects” to clean the previous results and then do the step1 again.

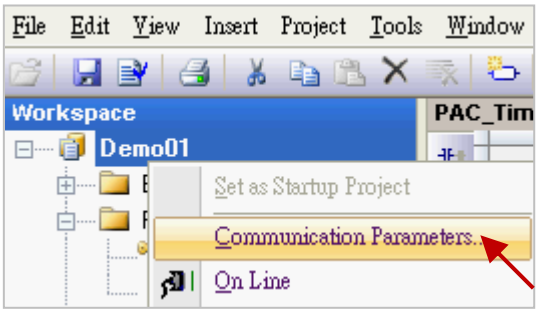




2.3.5 Download the Program to PAC

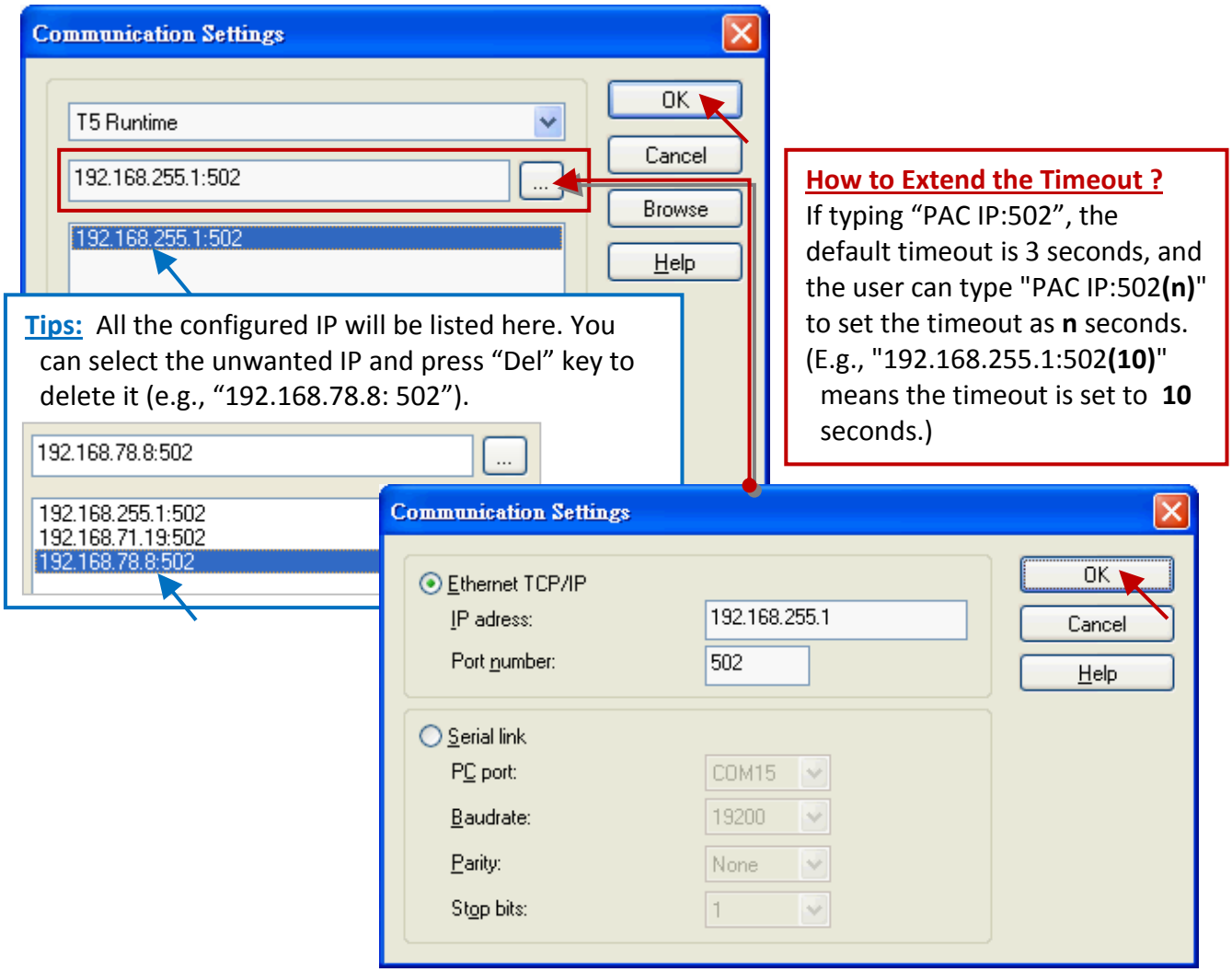
Before downloading the program, you need to set up the communication parameters. (By now, it only supports the Ethernet TCP/IP).

1. Mouse right-click the project name (i.e., "Demo01") and select “Communication Parameters…” to open the settings window.



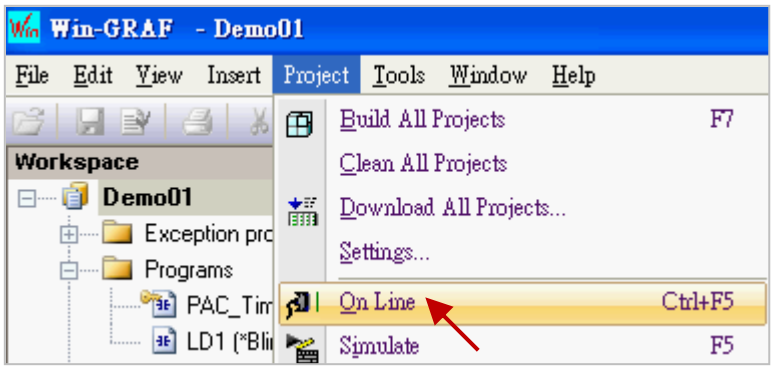
2. Enter the “PAC IP:502” (e.g., “192.168.255.1:502”) to add an IP address and then click “OK”.

   It can also click the button to add/modify the IP address.
   (Note: the default PAC IP is “192.168.255.1” and the fixed port number of Win-GRAF PAC is “502”)

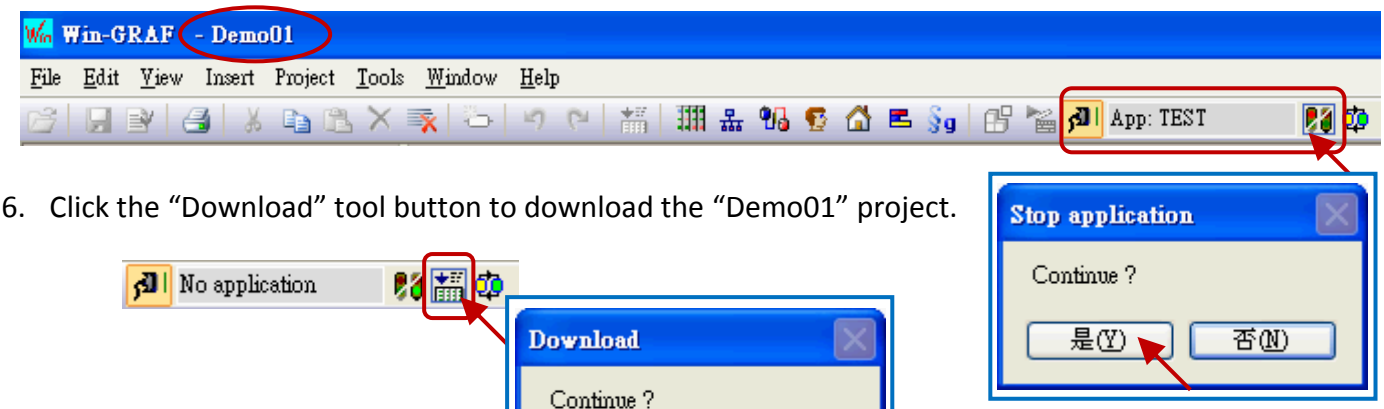


3. **Before establishing a connection, make sure the PAC and the network are working properly.**

4. Click the menu bar “Project” and select “On Line”, or click the  tool button to establish a connection.

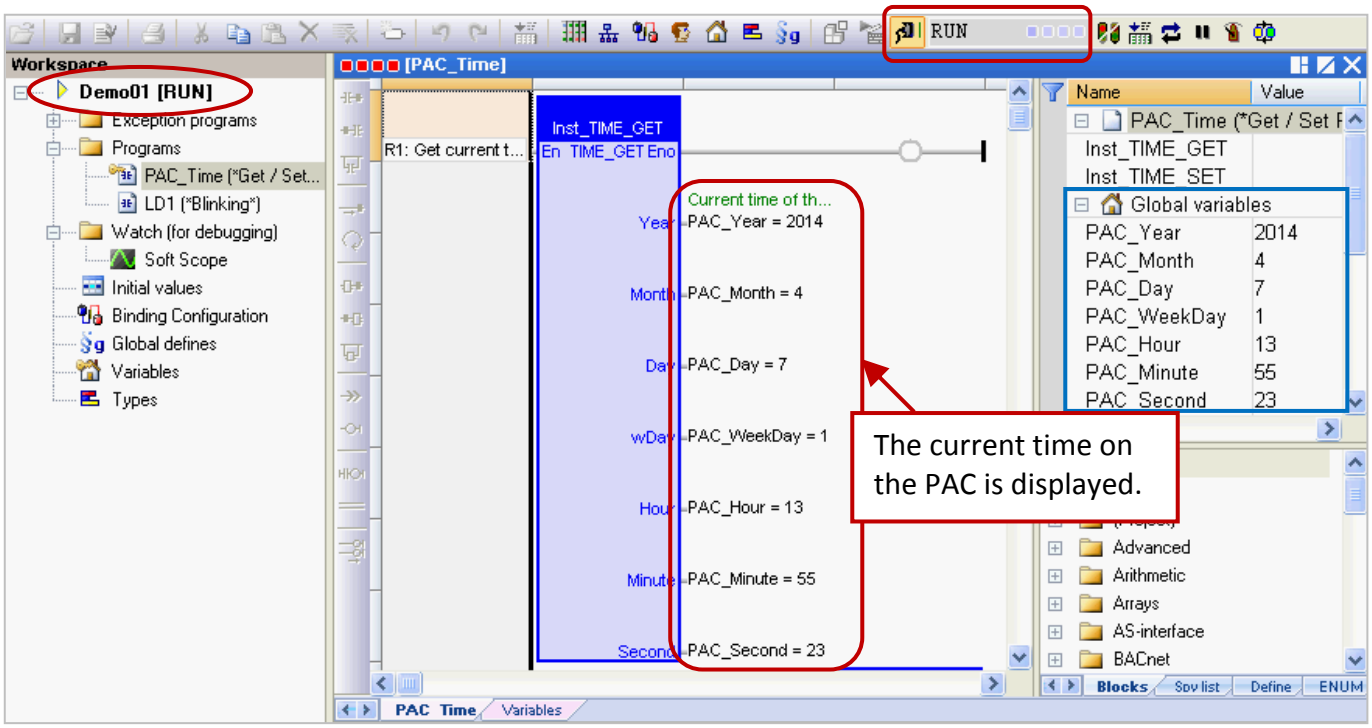


5. As the screenshot, if it shows “App: TEST”, different to the current project name (i.e., “Demo01”), that means there is a project (name: “TEST”) running on the PAC. Click “Stop application” tool button to stop the “TEST” project.



6. Click the “Download” tool button to download the “Demo01” project.

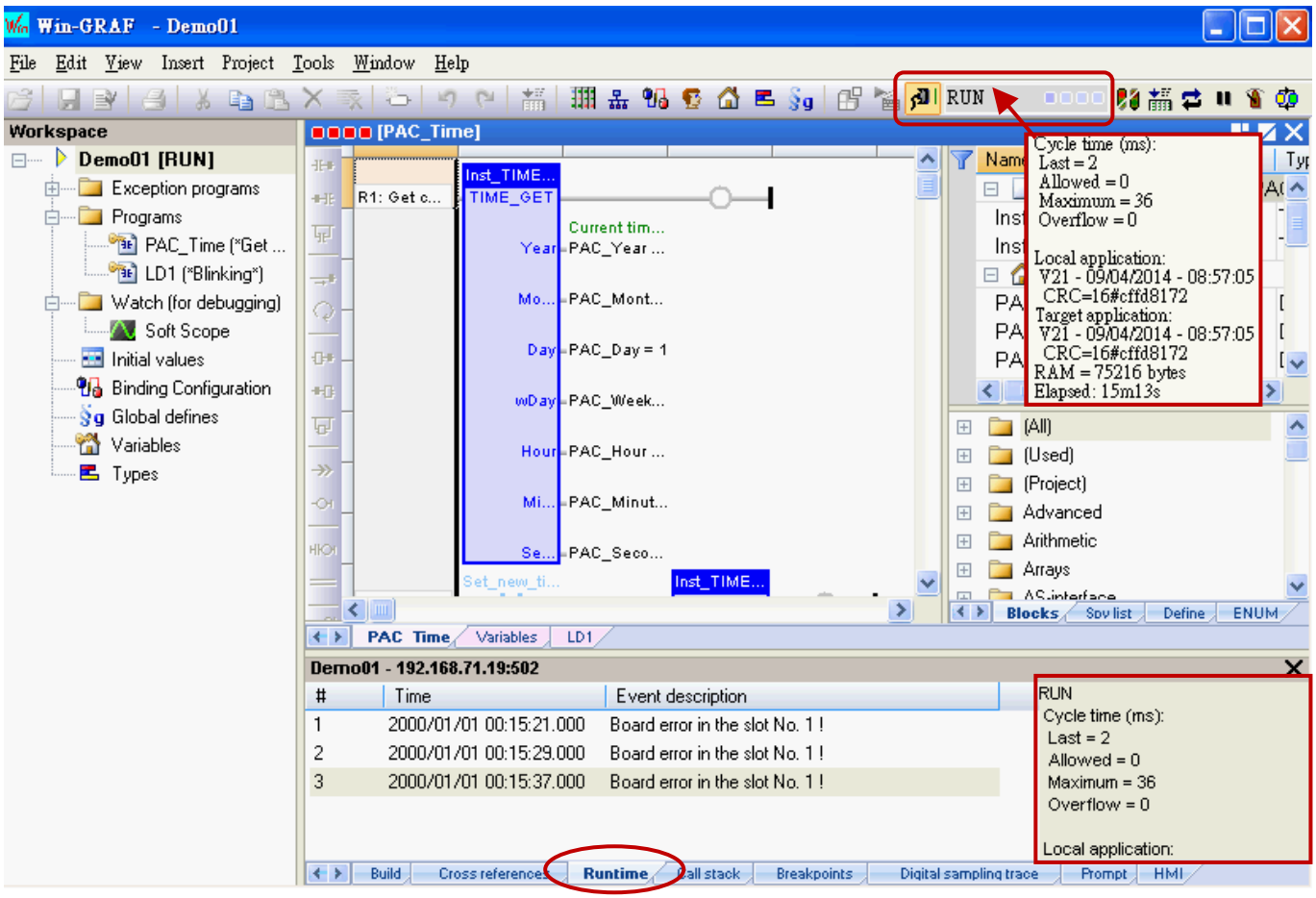
7. If RUN is displayed, it means that the "Demo 01" project has been successfully executed on the PAC.



Note: If there is any error message show up during the download process, refer the Appendix B to get the solution.

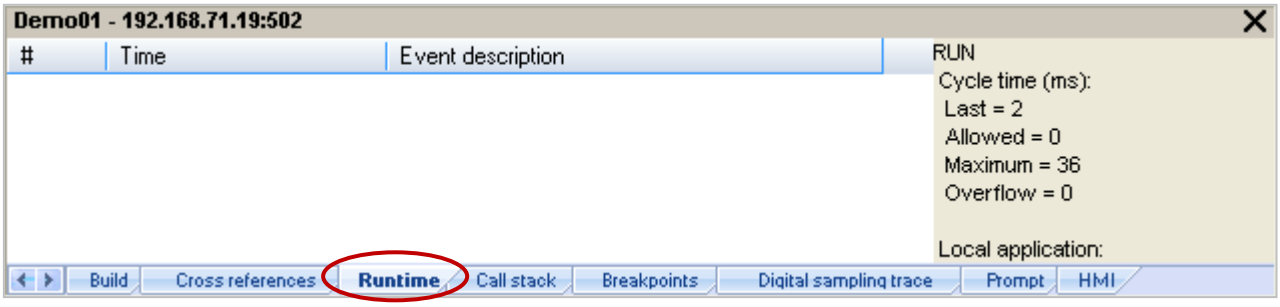
Cycle time

When on-line with the PAC, move your mouse over the "RUN" position on the toolbar to view the current cycle time of the program on the PAC. You can also view the cycle time in the bottom-right corner of the message area.



When doing the "On Line" ( ) operation, it will automatically switch to the "Runtime" tab and you can see if there is any error message for the downloaded program. (E.g., in this example, we need to plug the I-8055W module in the Slot1 of the PAC, and the message "Board error in the slot No. 1 !" means there is no I/O module in the Slot1 or an I/O exception.)

Shutting down the PAC, and plug one I-8055W module in the PAC's Slot1 then reboots. Then, click "On Line" ( ) button to connect to the PAC.

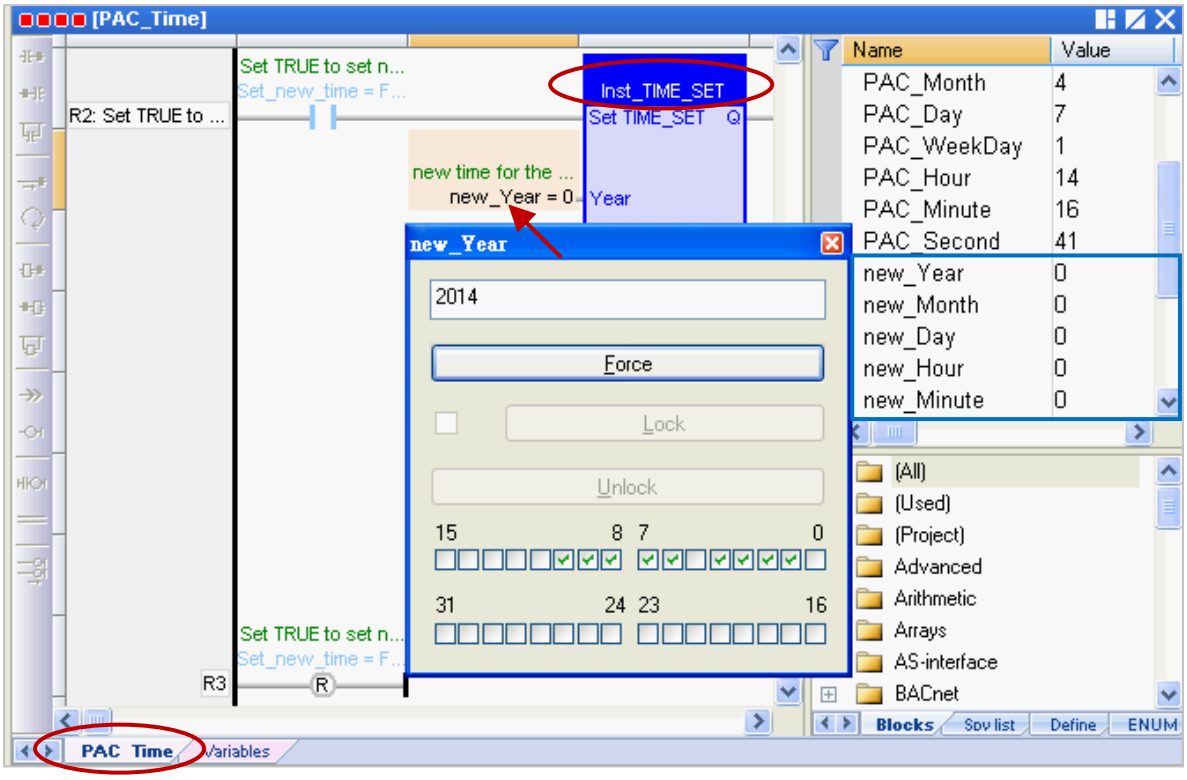


2.3.6 Testing the Program

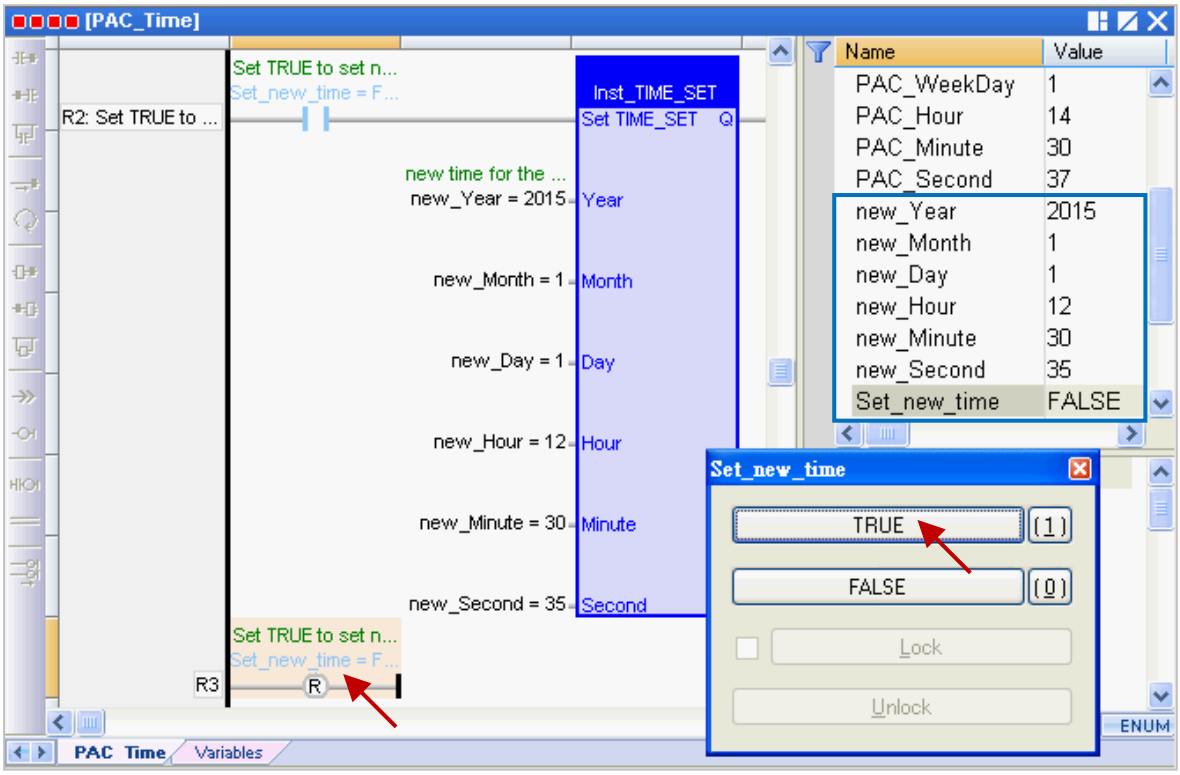
In the previous section, you have successfully downloaded the "Demo01" project and the following will describe how to test the program.

The "PAC_Time" Program:

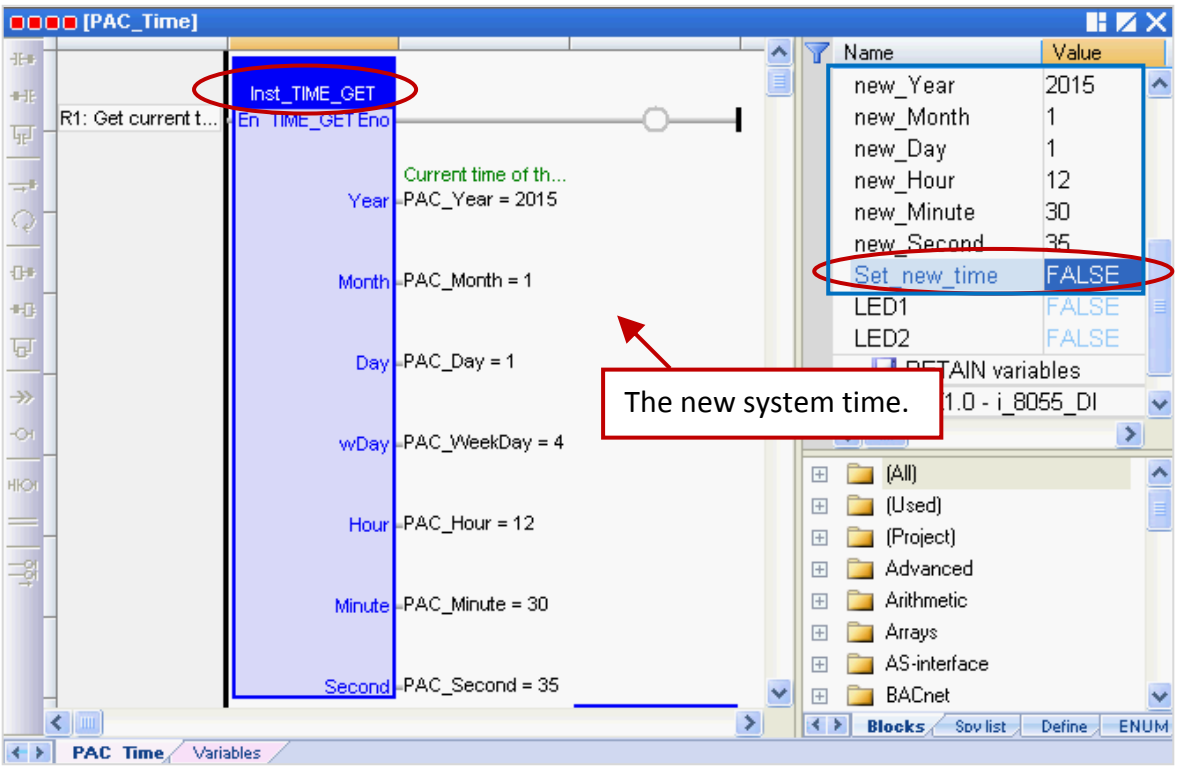
1. Mouse double-click the variable name (e.g., "new_Year") in the "TIME_**SET**" function block (or the Variables Area) one-by-one to change the PAC's system time (e.g., to change it as January 1, 2015 12:30:35).



2. Set the "Set_new_time" variable to "TRUE" to write the new system time.

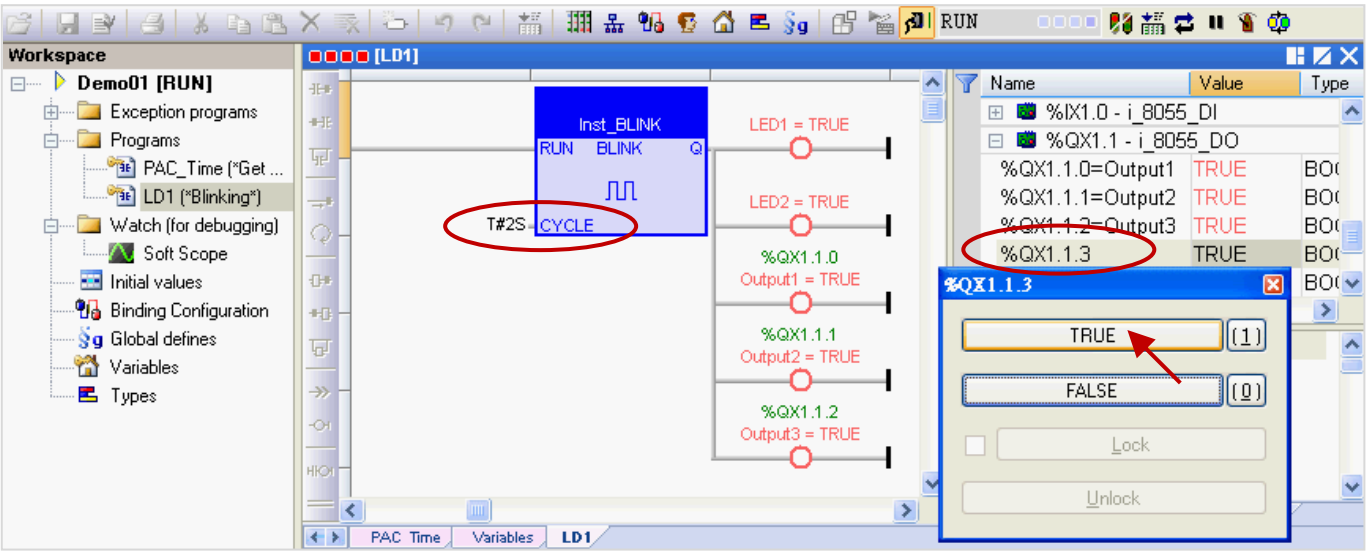


3. Then, the new system time will show on the “TIME_**GET**” function block (or the Variables Area), and the “Set_new_time” variable will be reset to “FALSE” automatically.



The “LD1” program:

4. When the “Demo01” project is running, you can check to see if the DO0, DO1 and DO2 tags of the I-8055W I/O module that plugged in the slot1 of the PAC is blinking every two seconds (like the value "T#2S” we set before). You can also assign a “TIME” variable on the left side of the “CYCLE” for easy to change the time setting. Refer the Section 2.3.1 for the setting way.
5. If the “%QX1.1.3” variable is set to “TURE” in the Variable Area, the LED4 (i.e., “DO3”, at the top of the I-8055W I/O module) will light up.



6. Click the  tool button again to cancel the PAC connection.

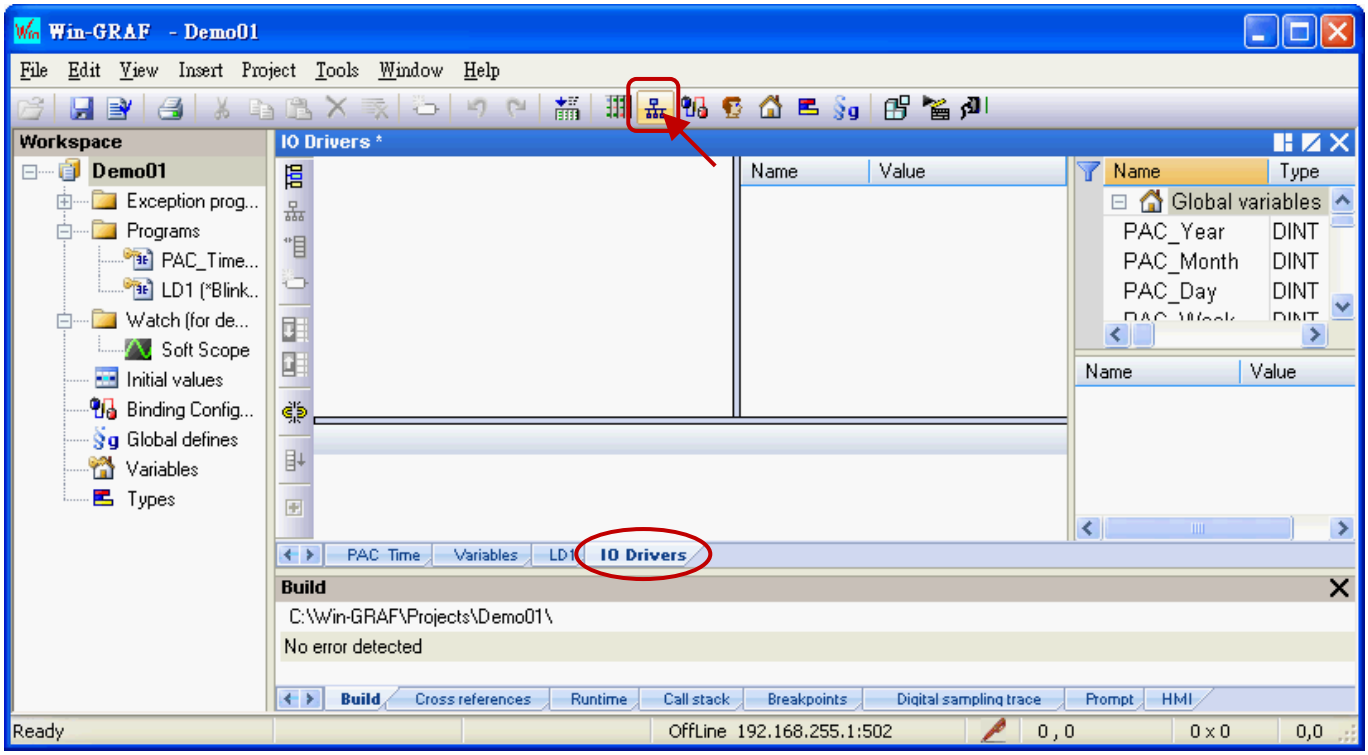
Note: Do NOT click the “Stop Application” button; it will stop the running project on the PAC.

Chapter 3 Modbus Slave: Allow the SCADA/HMI Software to Access Win-GRAF Variables

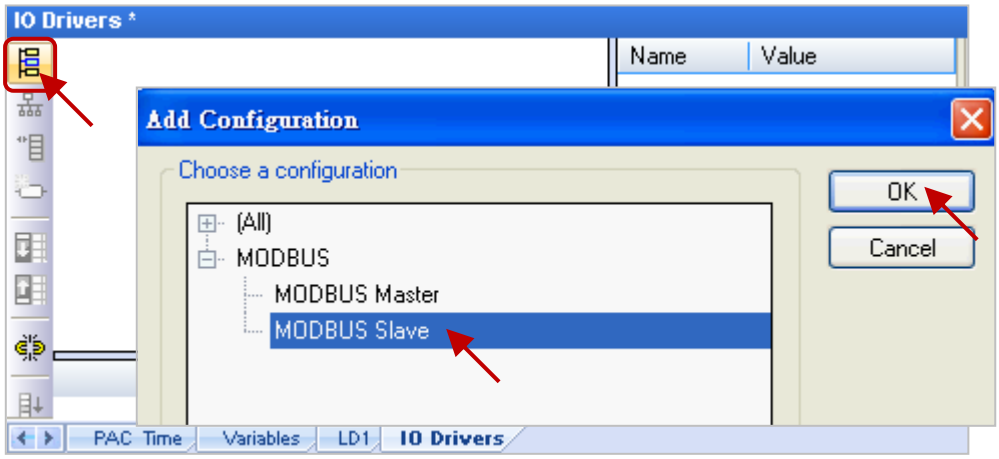
In Chapter2, we have described how to get/set the PAC system time (i.e., the “PAC_Time” program) and create a blinking function (i.e., the “LD1” program) in the “Demo01” project. The following sections describe how to allow the SCADA/HMI software (e.g., “InduSoft”) to access Win-GRAF variables that defined in the "Demo01" project. The Win-GRAF Workbench provides two ways to open the PAC data, one way is to enable the Win-GRAF PAC as a Modbus **TCP** Slave and the second way is to enable the Win-GRAF PAC as a Modbus **RTU** Slave (you must first complete all the Modbus Slave settings in Section 3.1, and then refer the Section 3.2). To begin, follow these steps:

3.1 To Enable the Win-GRAF PAC as a Modbus TCP Slave

1. Click the “Open Fieldbus Configuration” tool button to open the “IO Drivers” window.

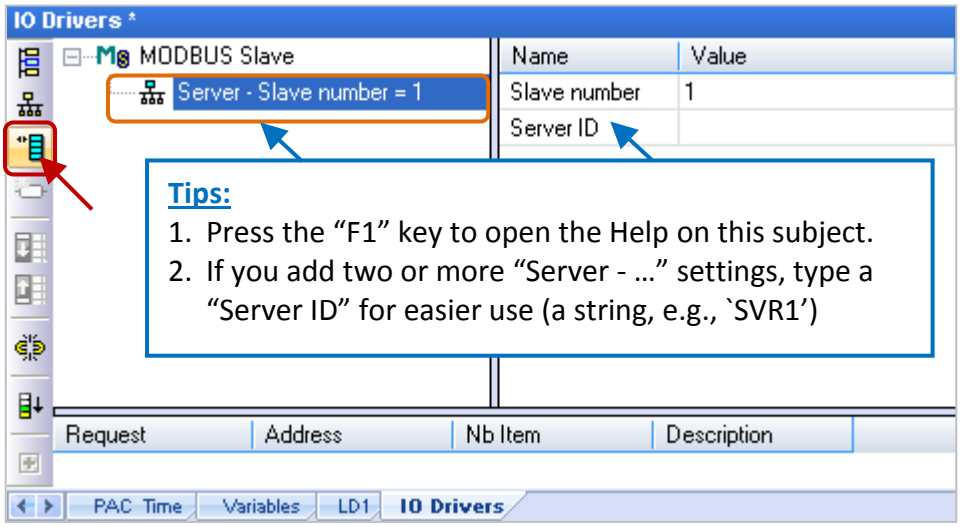


2. Click the “Insert Configuration” button on the left side of the “IO Drivers” window and then select the “MODBUS Slave” and click “OK” to enable a Modbus TCP Slave.

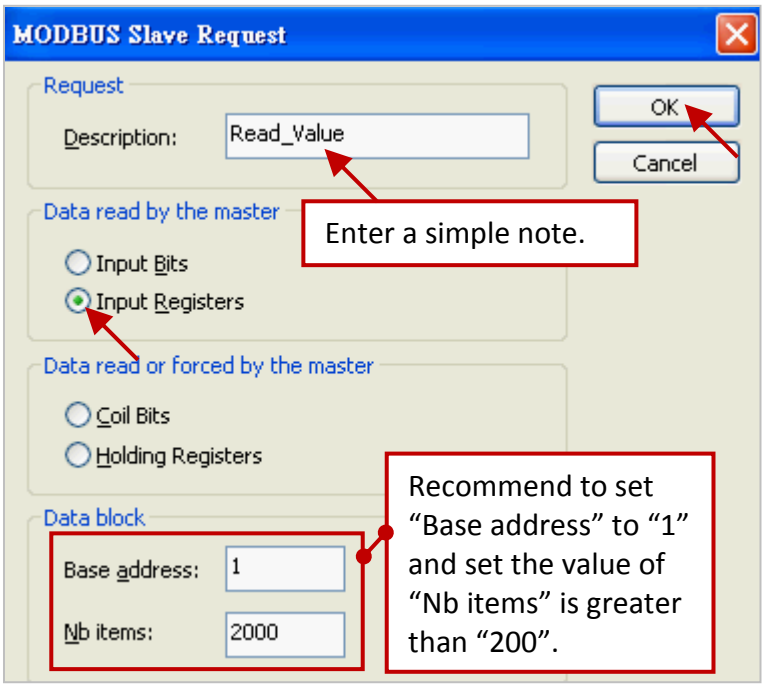


3. Click the “Insert Master/Port” button on the left side to set the “Slave number” (In this case, the value is “1”), and click the “OK” button.

4. Click the “Insert Slave/Data Block” button on the left side to open the “MODBUS Slave Request” window.



5. Enter a simple note in the “Description” field and then click the "Input Registers” option.



For Modbus Master to Read data:

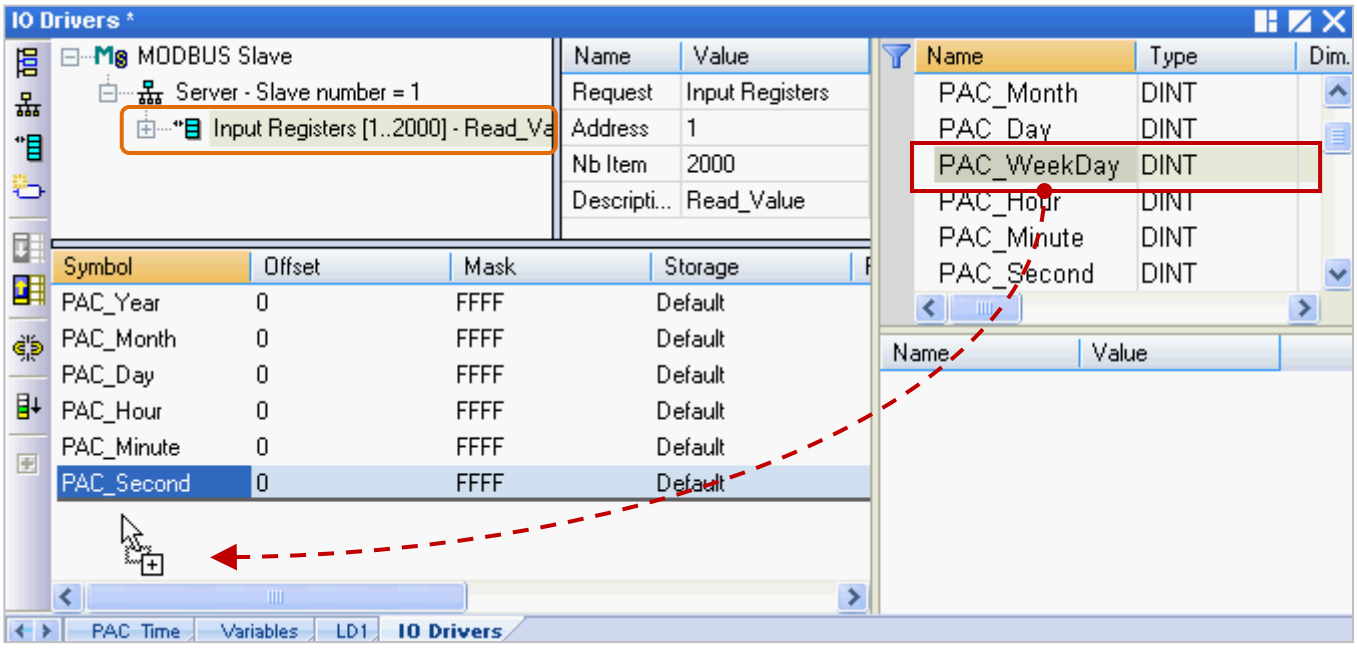
Options	Data types
Input-bits	BOOL
Input Registers	BYTE, INT, DINT, REAL, etc.

For Modbus Master to Write data:

Options	Data types
Coil-bits	BOOL
Holding Registers	BYTE, INT, DINT, REAL, etc.

(Refer the Appendix A to see more data type)

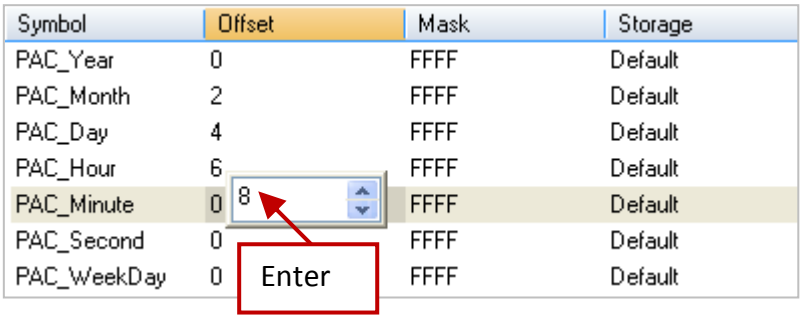
6. As the screenshot above, it's recommended to set the "Base address" to "1" and the "Nb items" refers to how much variable data can be provided by one "data block". If the data address requested from the Modbus Master (e.g., the SCADA software) is greater than this value (in this example, the value is "2000"), the Modbus Slave (i.e., Win-GRAF PAC) will not respond.
7. Mouse drags all the needed variables (e.g., "PAC_xxx", data type: "DINT") one-by-one from the Variables area and then drop it to the "Symbol" field.



8. Mouse double-click the "Offset" field and fill in a value, then press "Enter" key to finish the setting.

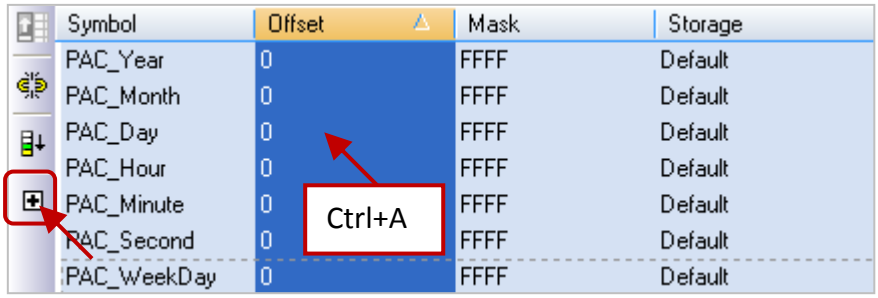
Note: (1) The "Offset" value starts at "0" and the Modbus address of variable is equal to this value plus 1 (Base address).

(2) If using a 32-bits (or more than 32-bits) data type (e.g., "DINT", refer the Appendix A), it requires two Modbus addresses, as the table below, the "Offset" values are 0, 2, 4, 6, etc.

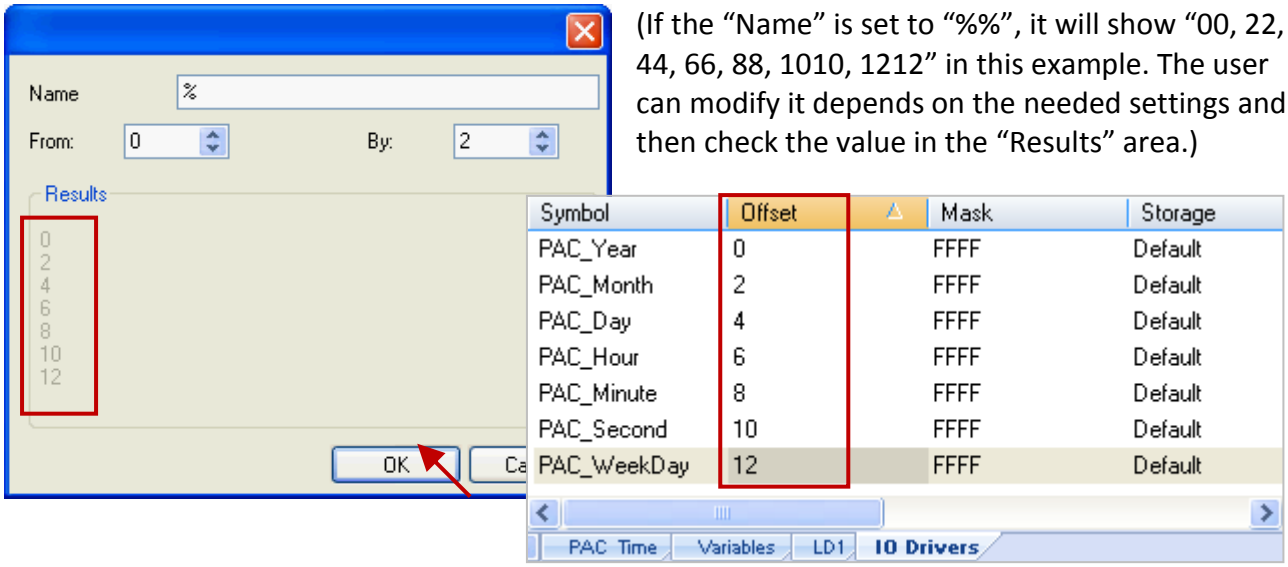


Tips:

Mouse click the "Offset" field and press the keyboard "Ctrl+A" to select all items, and then click the "Iterate Property" button at the left-side to open the settings window.

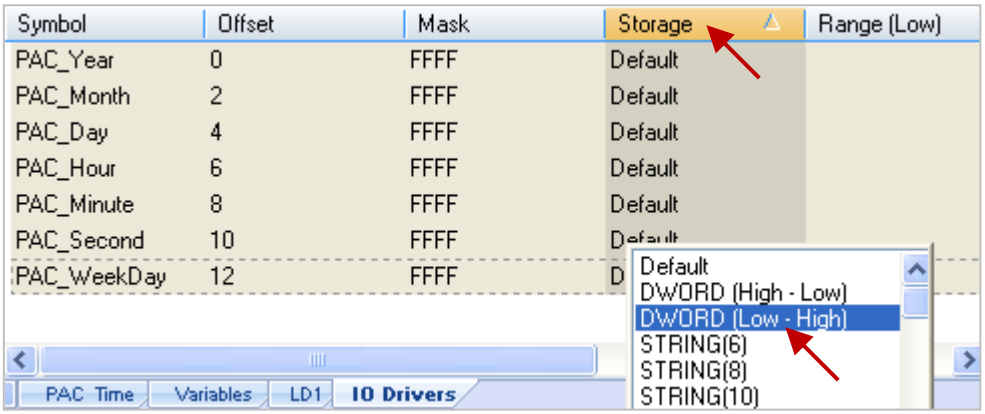


Keep the “Name” setting, enter “0” into “From” field and enter “2” into “By” field, then click “OK”.

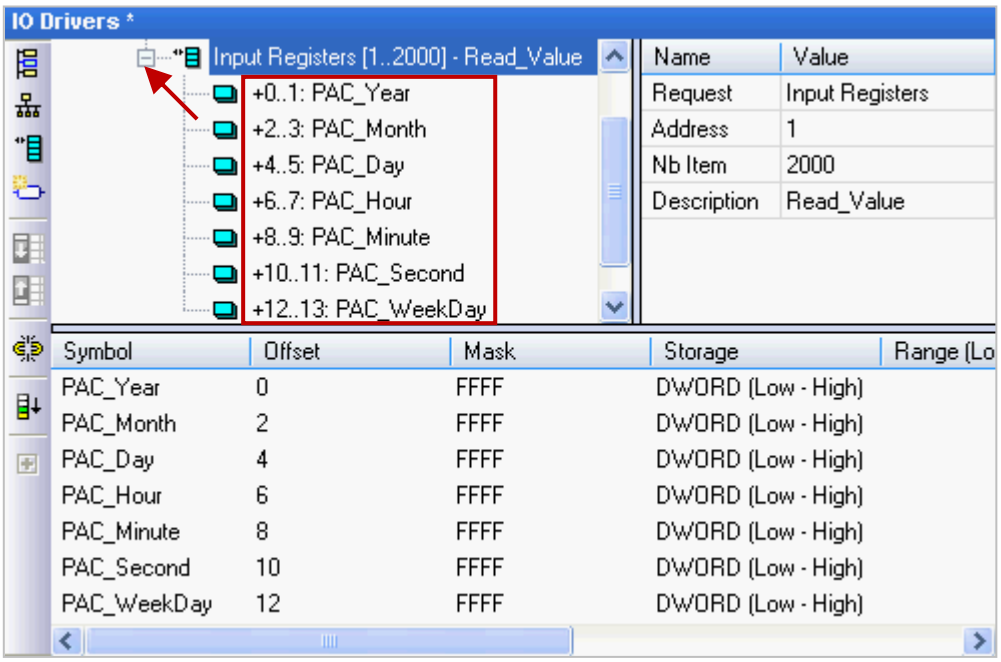


(If the “Name” is set to “%%”, it will show “00, 22, 44, 66, 88, 1010, 1212” in this example. The user can modify it depends on the needed settings and then check the value in the “Results” area.)

9. Click “Storage” to select entire columns and then press “Enter” key to display a drop-down menu. Then, select “DWORD (Low – High)” and press “Enter” key to complete the setting. (If using a 16-bits or below, it’s no need to set the “Storage” item.)



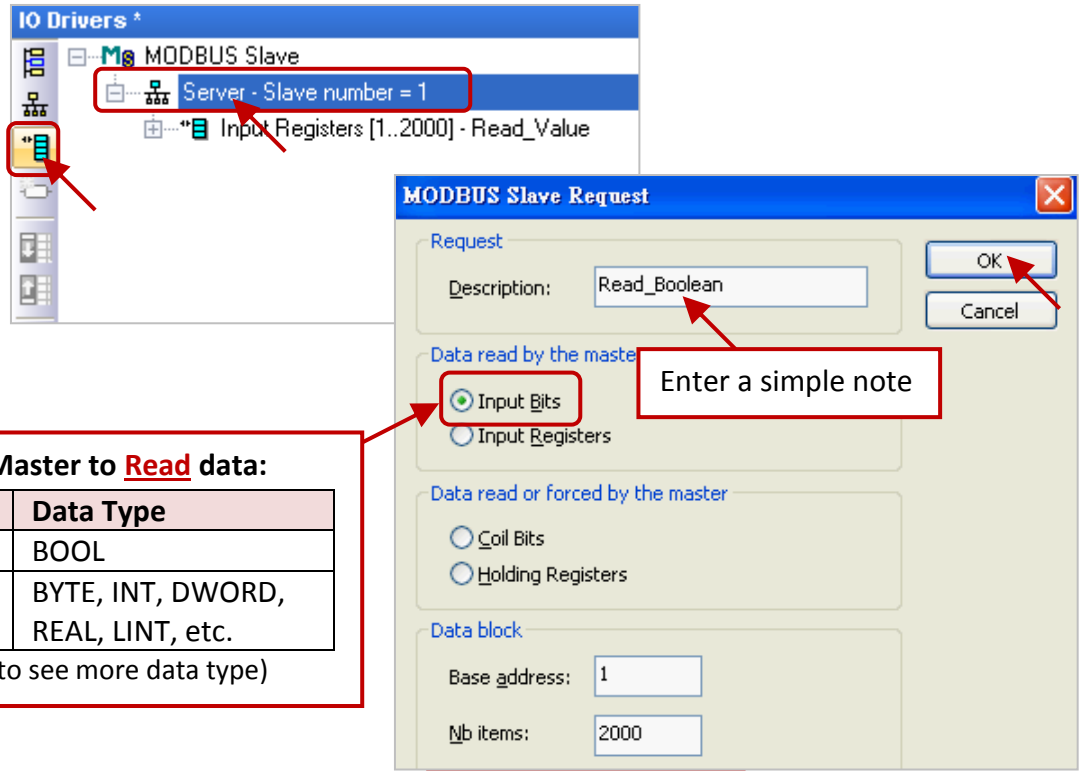
To expand this “Data Block” and you can see the Modbus addresses of all variables. It equals to the “Offset” value plus 1 (Base address).



10. Then, we need to add the second “Data Block” for the Modbus Master to read the Boolean data. This configure way is similar to the step 4 to 8:

(1) Click the “Server - …” item and click the “Insert Slave/Data Block” button at the left side to open the settings window.

(2) In the “MODBUS Slave Request” window, enter a simple note and select the “Input-bits” option, then set “Base address” to “1” and set “Nb items” to “2000”.

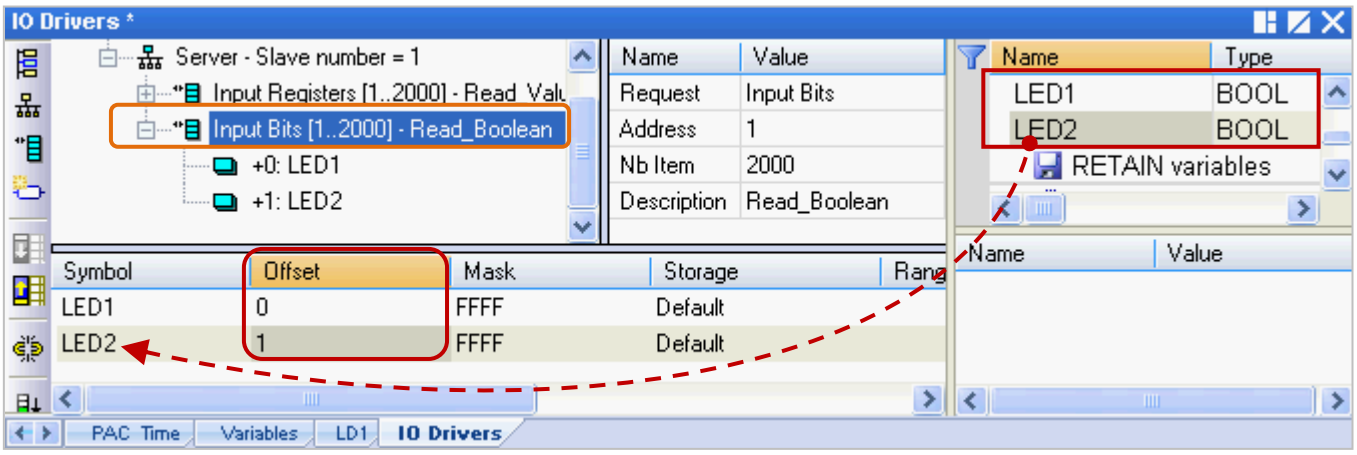


For the Modbus Master to Read data:

Option	Data Type
Input Bits	BOOL
Input Registers	BYTE, INT, DWORD, REAL, LINT, etc.

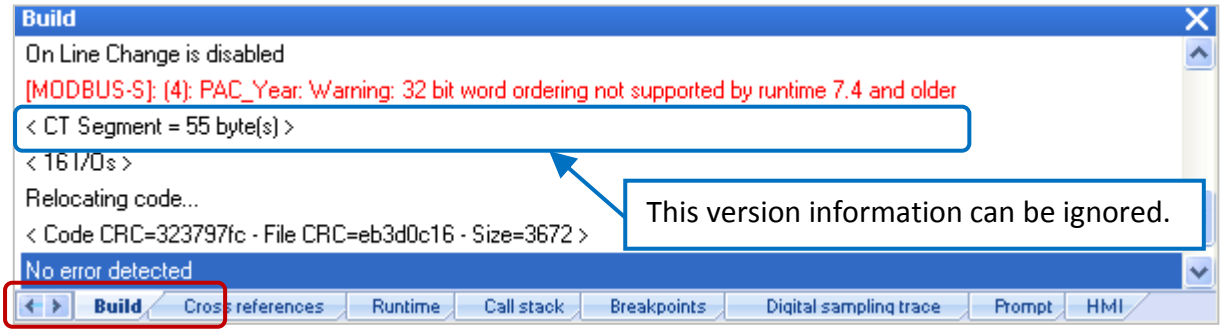
(Refer Appendix A to see more data type)

(3) Mouse drags the Boolean variables (i.e., “LED1”, “LED2”; data type: BOOL) one-by-one and drop them to the “Symbol” area, and then set the “Offset” to “0” and to “1”.

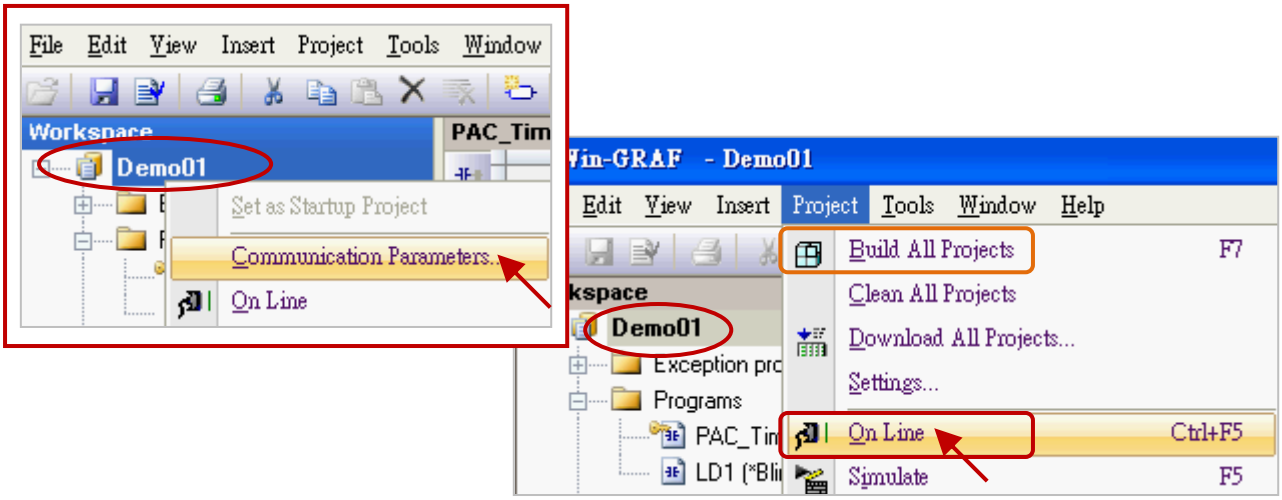


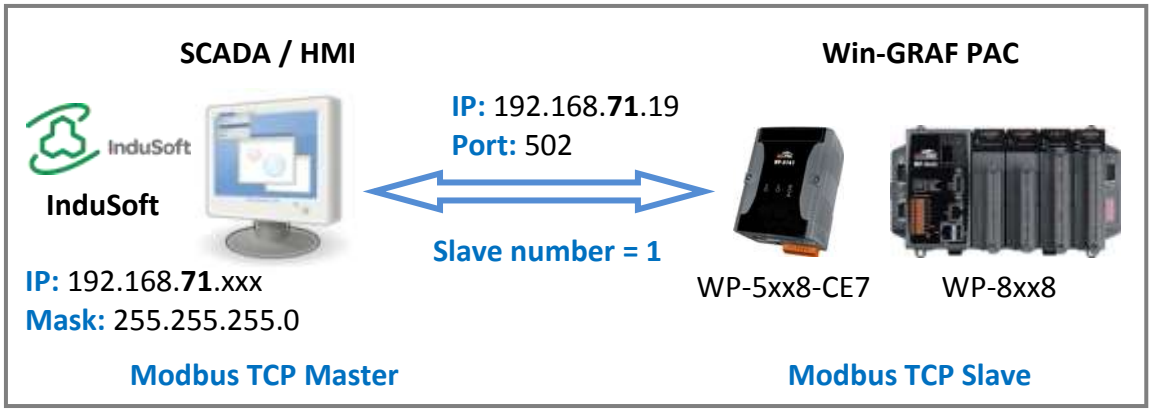
You have completed the settings for the Modbus Slave. Finally, follow the way below to re-compile the program and download it to the Win-GRAF PAC.

11. Click “Project” > “Build All Projects” from the menu bar to compile this program again (refer the Section 2.3.4). If a message informs you “No error detected” that means this process is successful.



12. Mouse right-click the project name (i.e., “Demo01”) and select the “Communication Parameters…” to set the PAC IP (e.g., “192.168.71.19:502”) and then click the menu bar “Project” > “On Line” (or) to establish a connection and download this project to the Win-GRAF PAC. (Refer the Section 2.3.5).





(Refer the P1-1 to view all PAC models)

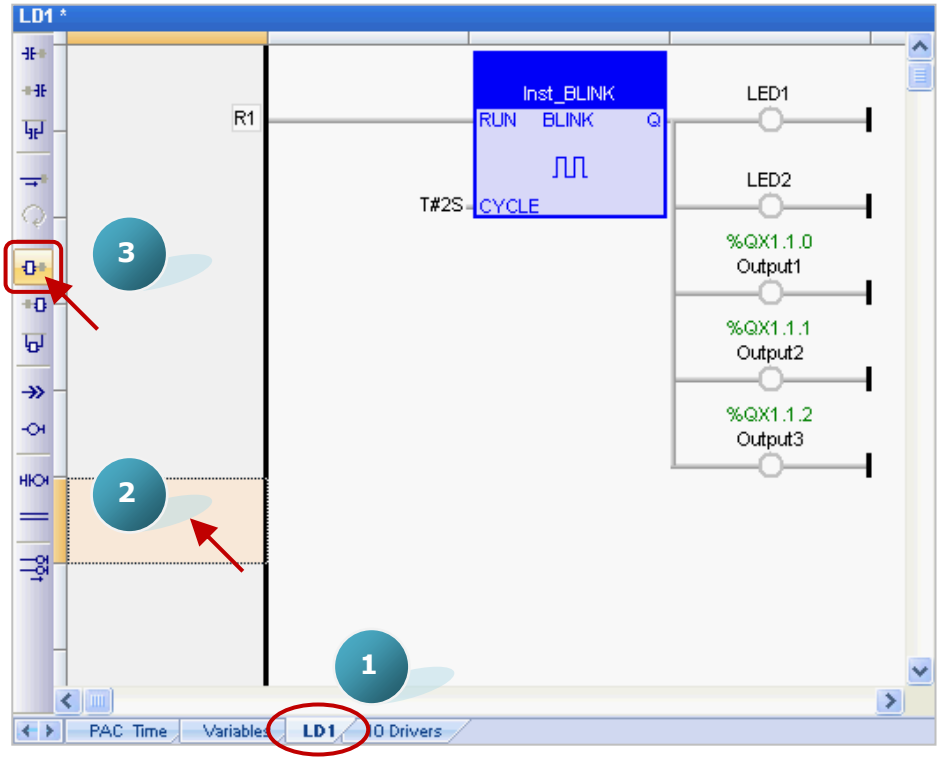
After completing all the steps, the HMI/SCADA software can access to all the Win-GRAF variables listed above via Modbus TCP protocol.

3.2 To Enable the Win-GRAF PAC as a Modbus RTU Slave

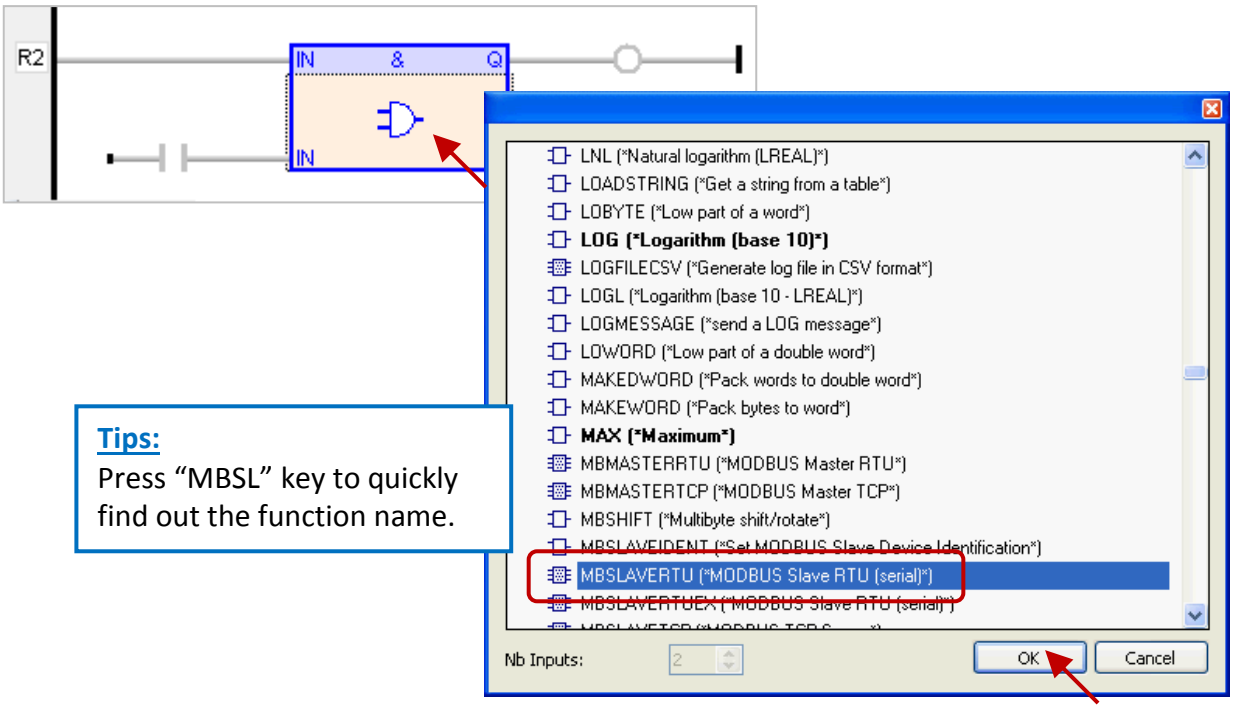
Before doing this, you must complete all the content that described in Section 3.1 to open the Modbus Slave data. The way to enable the Win-GRAF PAC as the Modbus RTU Slave is to add the "MBSLAVERTU" or the "MBSLAVERTU**EX**" function block in the program. To begin, follow these steps:

Add the "MBSLAVERTU" function block

1. In the "LD1" window, mouse click the place where you want to add this function block and then click the "Insert FB.." button on the left side of the window.

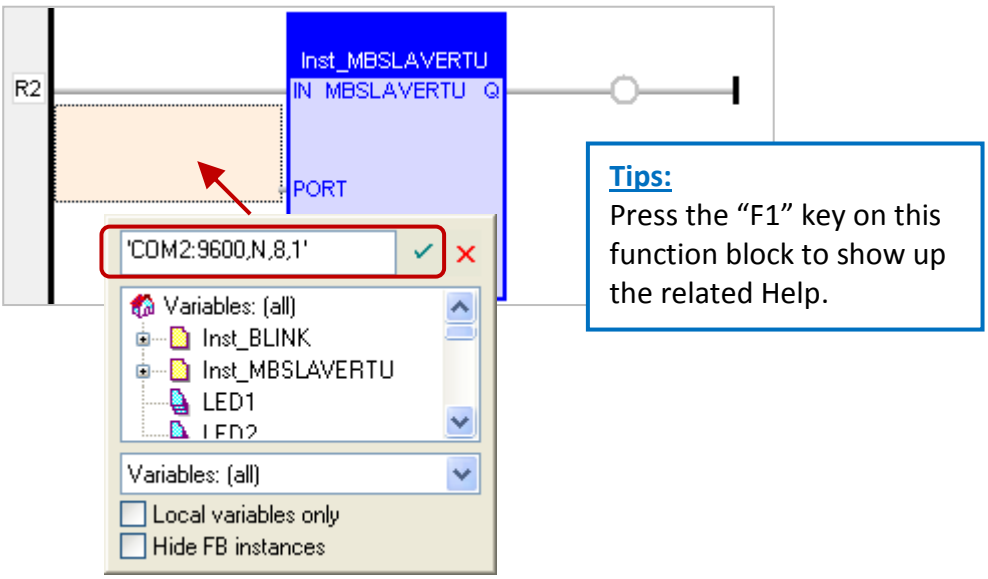


2. Double-click on this function block and select the name "MBSLAVERTU", then click "OK".

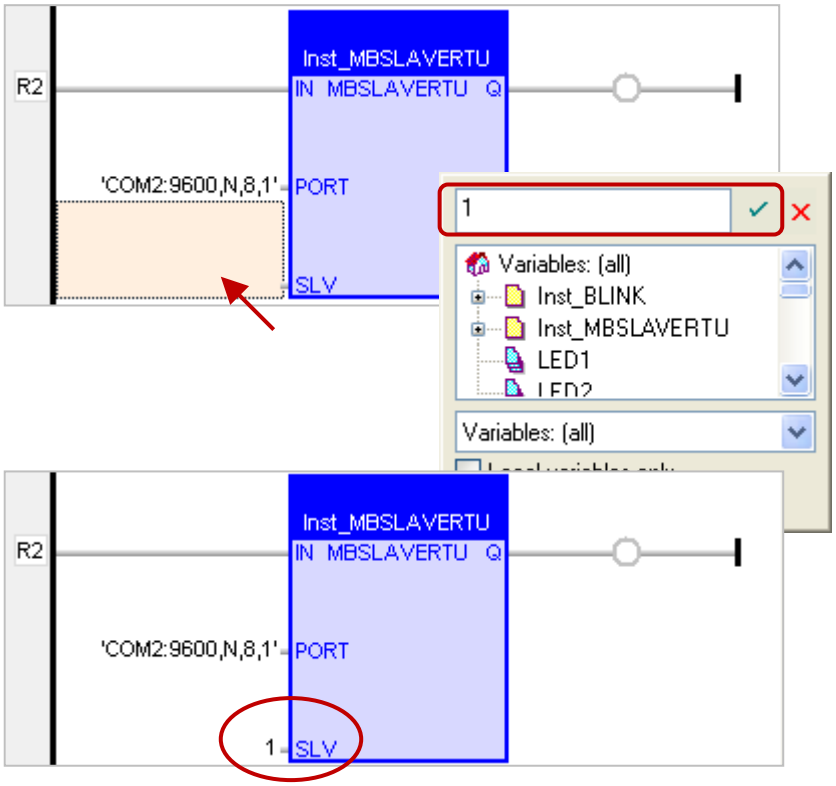


Tips:
Press "MBSL" key to quickly find out the function name.

3. In the “MBSLAVERTU” function block, mouse double-click the left side of the “PORT” and enter a string 'COM2:9600,N,8,1' (it means using the Win-GRAF PAC’s COM2 to communication with the Modbus Master) and then click ✓ to complete the settings.

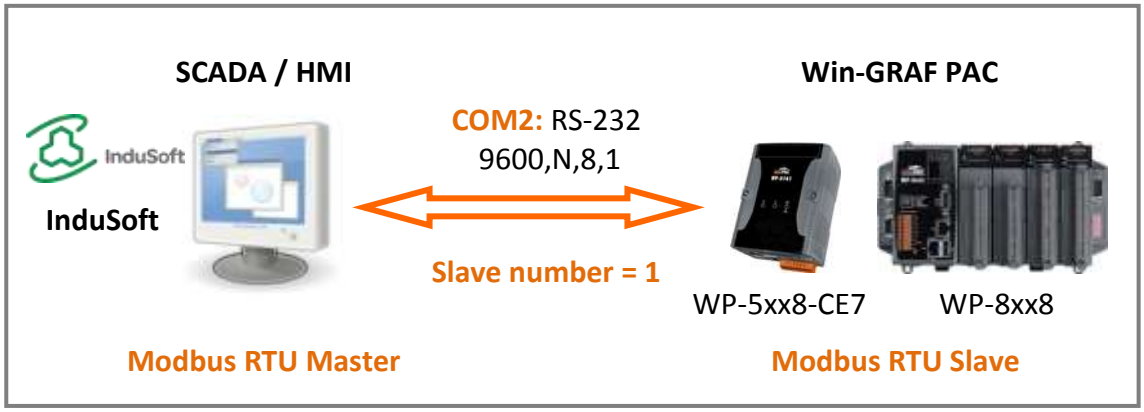


4. Double-click the left side of the “SLV” and then enter “1” (the value set in the Section 3.1 - Step 3), then click ✓ to finish the setting.



Now, you have completed the setting of the “MBSLAVERTU” function block, and then re-compile the program and download it to the Win-GRAF PAC. (Refer the Section 2.3.4, Section 2.3.5)

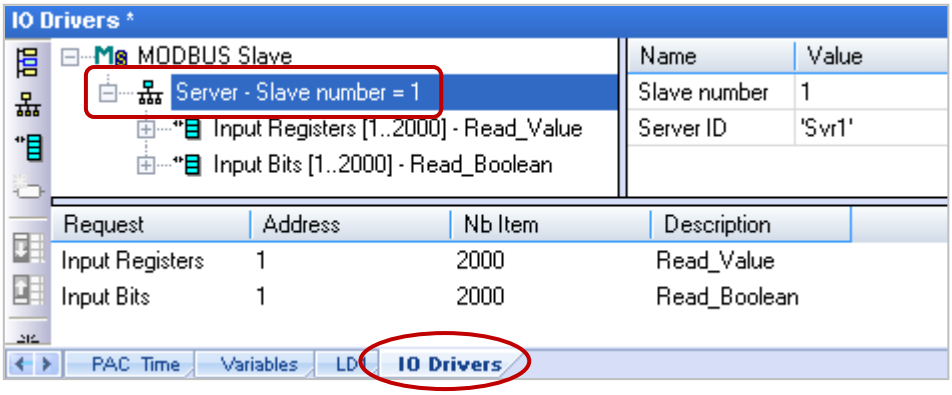
Note: Users can enable multiple Modbus RTU Slave ports for each PAC (recommend not over 16 Ports), the way is to add multiple “MBSLAVERTU” function blocks and set the different “Port” value.



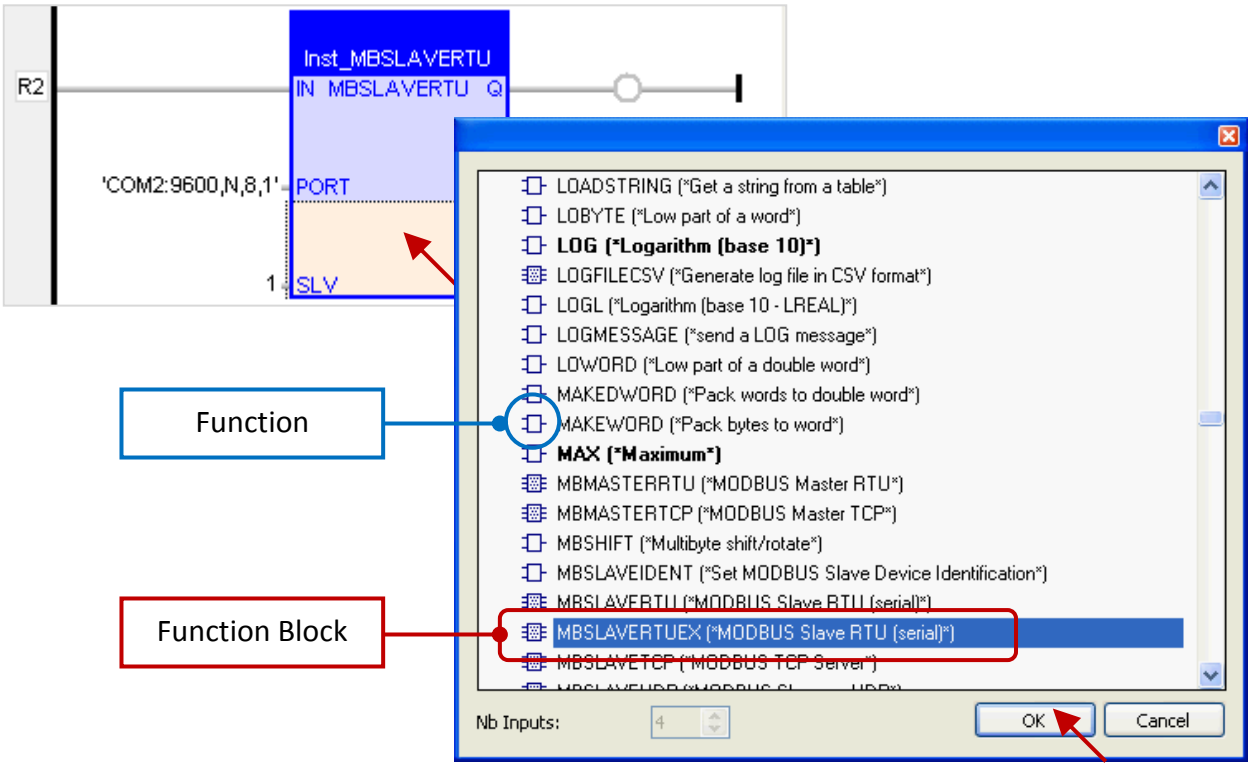
(Refer the P1-1 to view all PAC models)

Add the "MBSLAVERTUEX" function block

When using several "Server - …" settings (recommend to set one) in the "IO Drivers" window, the user needs to use the "MBSLAVERTU**EX**" function block.



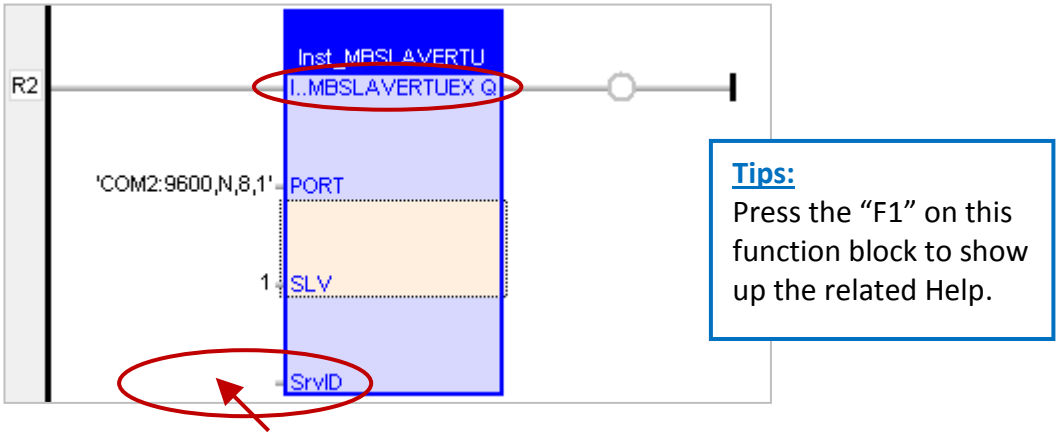
1. Follow steps 1 to 4 above to add the "MBSLAVERTU**EX**" function block. If you want to change the usage for existing function block, mouse double-click the "MBSLAVERTU" and change it to the"MBSLAVERTU**EX**", and then click the "OK" button.

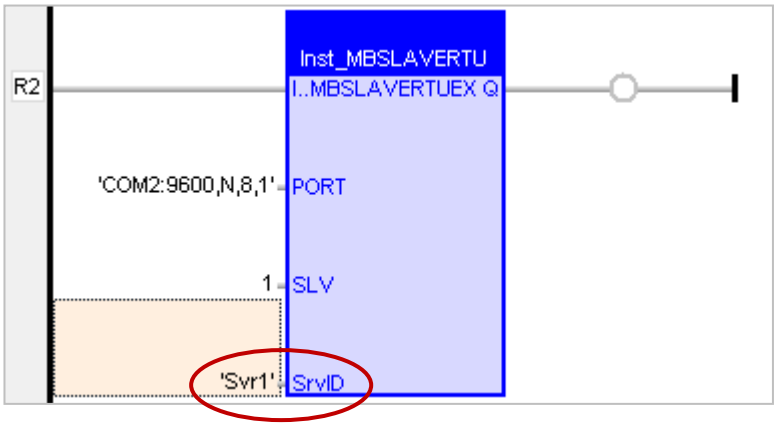


2. The “MBSLAVERTU**EX**” has a “SrvID” setting. Double click the left side of the “SrvID” and enter a needed “Server ID” (using a string format, e.g., ‘Svr1’).

   Note:

   Using the “MBSLAVERTU” function block means the first Modbus Slave setting will be enabled. Using the “MBSLAVERTU**EX**” function block means to enable the Modbus Slave setting depends on the “Server ID”.





Now, you have finished the settings for the “MBSLAVERTU**EX**” function block, and then re-compile the program and download it to the Win-GRAF PAC. (Refer the Section 2.3.4, Section 2.3.5)

Chapter 4 Linking "I/O Boards"

This section lists the usage of the "I/O Boards" function in the Win-GRAF Workbench to link the Real I/O modules or to enable other I/O functions. First, you need to know the slot numbering and supported I/O modules for each PAC:

PAC Model	Slot No. (from the left to the right)	The supported PAC I/O modules
WP-8xx8	0 to 7	**Supported:** I-8K and I-87K series (High Profile) I/O modules. (E.g., I-8017HW and I-87055W) **Not Supported:** I-8K and I-87K series (Low Profile) I/O modules. (E.g., I-8017H and I-87055)
WP-8xx8-CE7	0 to 7	
XP-8x38-CE6 (*)	0 to 7	
XP-8x48-CE6 (*)	1 to 7	
VP-x2x8-CE7 (*)	0 to 2	
WP-5238-CE7	-	**The Palm-size PAC** which can support one XV board. (E.g., XV107, XV116, XV308, etc.) The XV board is a kind of the Modbus slave I/O board. (Refer the Section 5.1.6 to 5.1.12 for using XV-boards.)

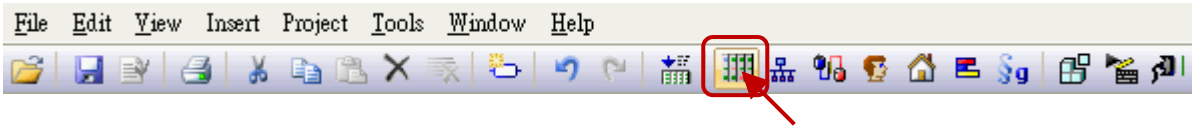
(*) : The XP-8038-CE6, XP-8048-CE6, VP-x208, WP-5xx8-CE7 are the 0-slot PAC.
(Refer P1-1 for all PAC models)

Add the "I/O board"

"I/O board" refers to the I/O functionality in the Win-GRAF (e.g., "i_8037_DO") and "I/O module" refers to the hardware device (e.g., "I-8037W").

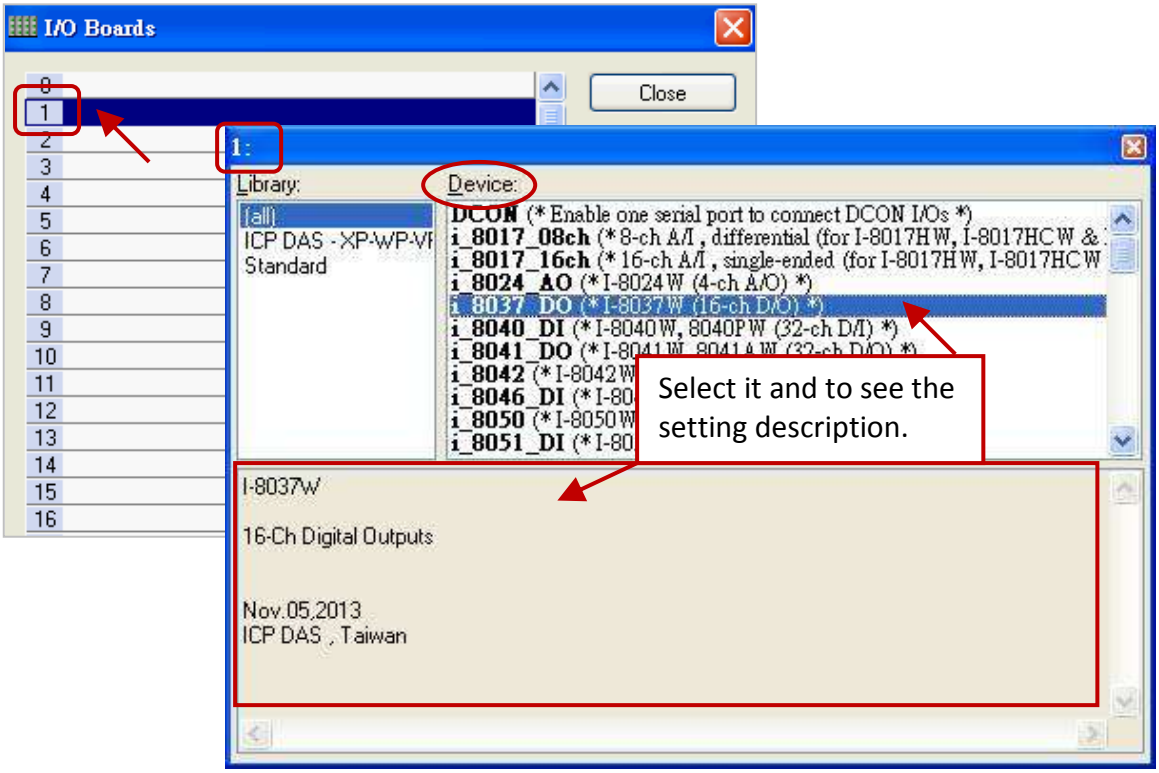
Notice:
Besides the software setting for the I/O board, there are some kinds of I/O modules need to set the hardware Jumper (e.g., Single-ended and Differential Jumper). So, go to the website to look up the product information, or the description printed on the module cover, or the attached shipment document.
I-8K and I-87K series product website:
http://www.icpdas.com/root/product/solutions/remote_io/rs-485/i-8k_i-87k/i-8k_i-87k_selection.html

1. In the Win-GRAF, click the "Open I/Os" button from the toolbar to open the "I/O Boards" window.

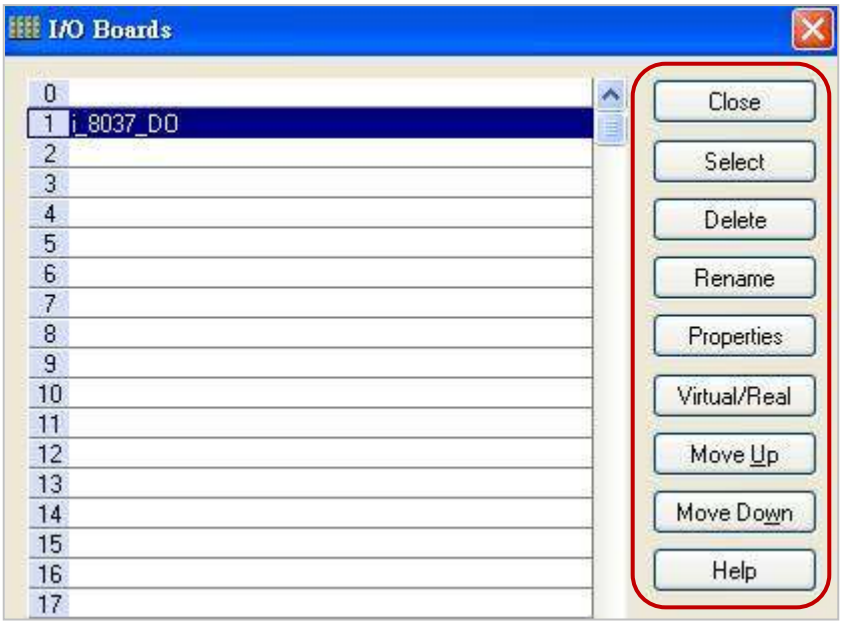


2. As the figure below, mouse double click the slot number that corresponds to the real I/O module and then select the proper I/O board (e.g., i_8037_DO).

Note: The Slot 0 to Slot 7 are reserved for real I/O modules that plugged into the PAC, and the slot 8 or above are for other usage.



<u>Buttons Description</u>: (Click the following buttons to modify the settings)



"Close":
Close this window.

"Select":
Open the I/O selecting window.
(Hot Key - "Enter": to open ;
Hot Key - "ESC" : to exit)

"Delete":
Delete this I/O board.

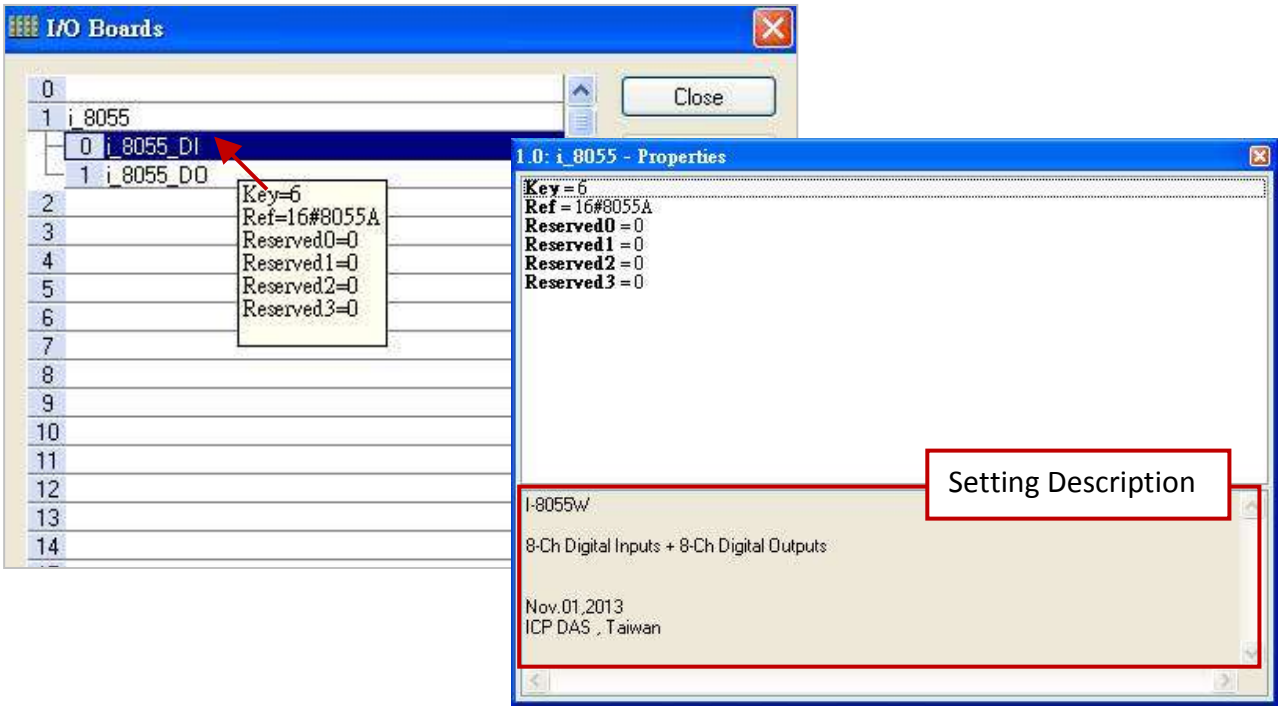
"Rename":
Rename this I/O board.

"Properties":	Look up the usage of this I/O board.
"Virtual/Real":	Swith the I/O board to a Virtual I/O (for testing) or a Real I/O. (Hot Key – "Space")
"Move Up":	Move up this I/O board.
"Move Down":	Move down this I/O board.
"Help":	To see the description on "I/O devices".

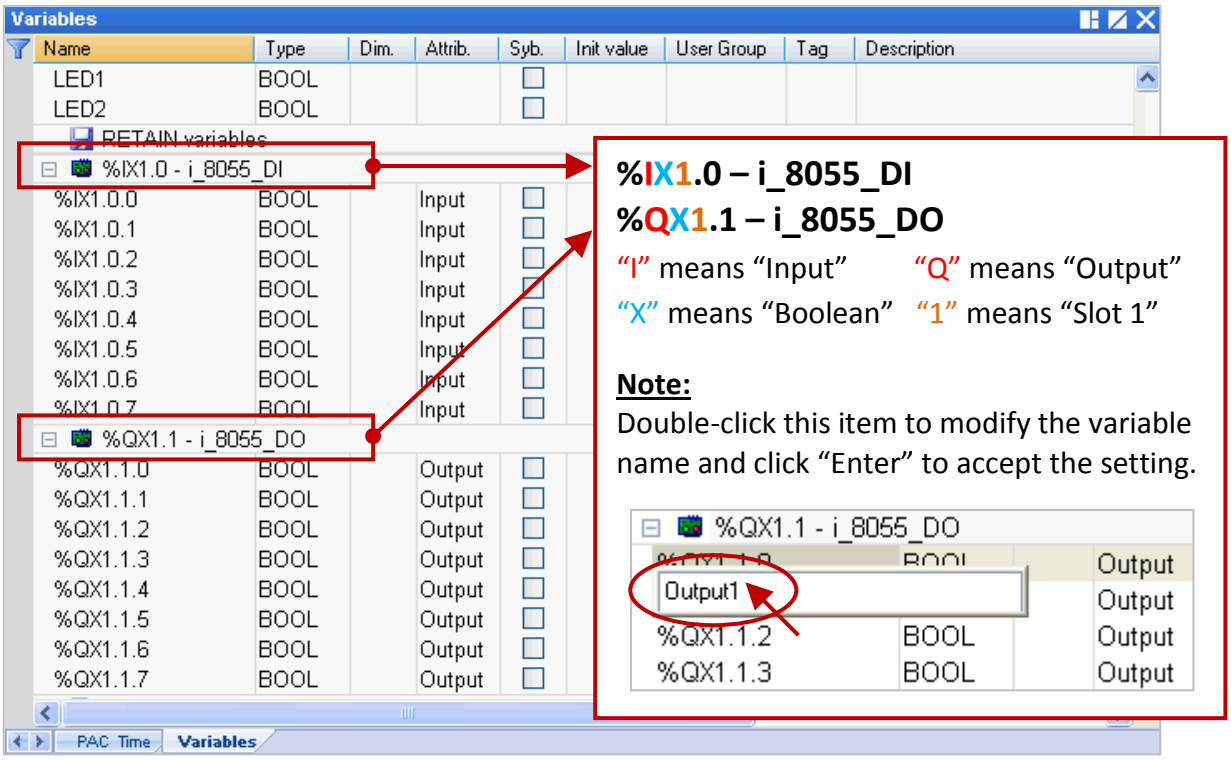
4.1 DI/DO Boards

Here use the "I-8055W" as an example, users can refer the Chapter 4 (P4-1) to add this I/O board.

1. Double-click the “i_8055_DI” (or the “i_8055_DO”) to open the “Properties” window.
   Note: A mouse-over showing the details on the “i_8055_DI” (or the “i_8055_DO”).



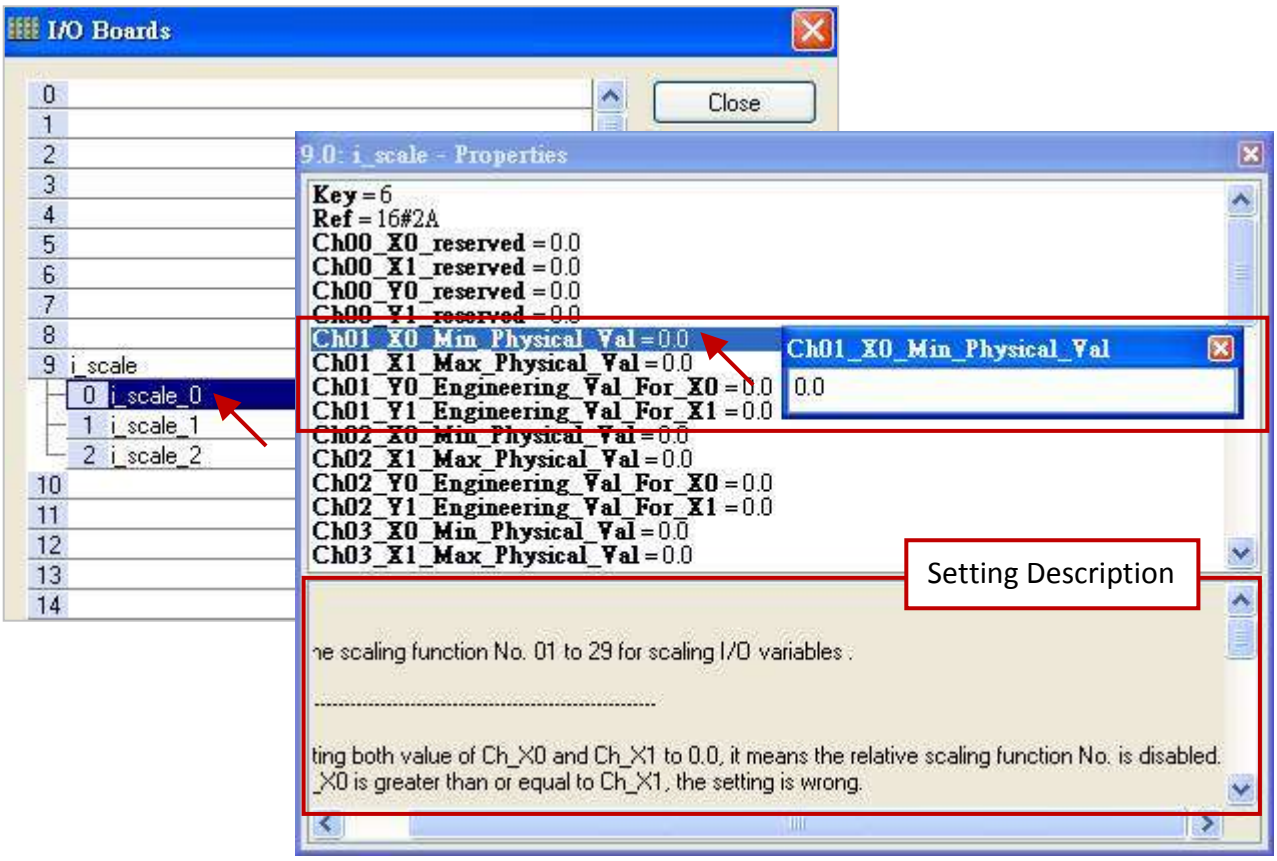
2. After linking the “i_8055” I/O board, it will auto add 8 Input and 8 output variables in the “Variables” window that can be used in the program.



4.2 i_scale (Conversion Table)

The “i_scale” function can set up to 29 scaling functions to convert values for the AI or AO module that plugged in the slot 0 to slot 7. See the Chapter4 (P4-1) to add this I/O board.

1. Mouse double click the "i_scale_0" (or "i_scale_1" or "i_scale_2") to open the “Properties” window, and then to see the setting description.
   Note: Using the slot 8 or above No. because the slot 0 to slot 7 are reserved for the real I/O module.



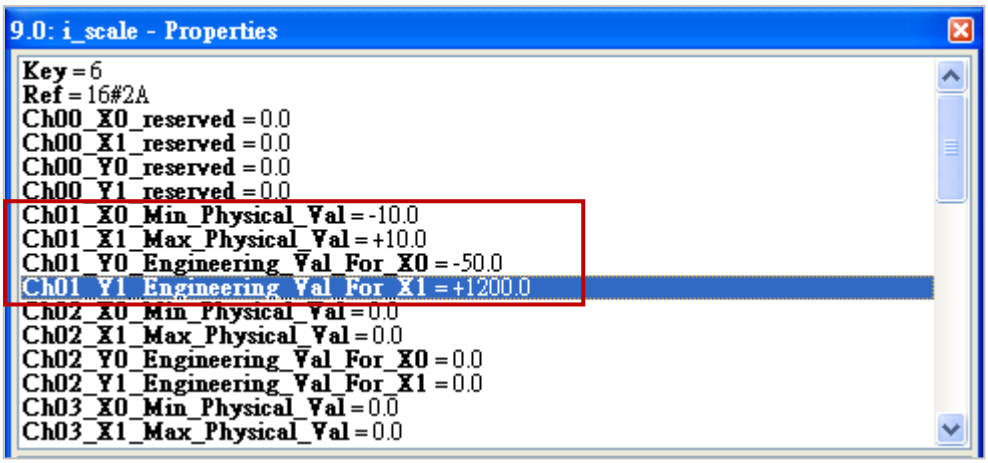
Parameters: (“Ch” means the Ch01 to Ch29, "Ch00” is a reserved item)

Ch_X0_Min_Physical_Val:	The min. value of AI (or AO) boards (X0).
Ch_X1_Max_Physical_Val:	The max. value of AI (or AO) boards (X1).
Ch_Y0_Engineering_Val_For_X0:	The engineering value after scaling X0.
Ch_Y1_Engineering_Val_For_X1:	The engineering value after scaling X1.

2. Double click the item and fill in a value, then press the “Enter” key to complete the setting.
   Notice:
   1. If set both Ch_X0 and Ch_X1 values to "0.0", it means the relative scaling function No. is disabled.
   2. If Ch_X0 is greater than or equal to Ch_X1, the setting is wrong.
   3. If Ch_Y0 is equal to Ch_Y1, the setting is wrong.

   For example, if the AI board's value is 4 to 20 mA and wish to scale as 0 to 10000, then set Ch_X0 as "4.0", Ch_X1 as "20.0", Ch_Y0 as "0.0", Ch_Y1 as "10000.0".

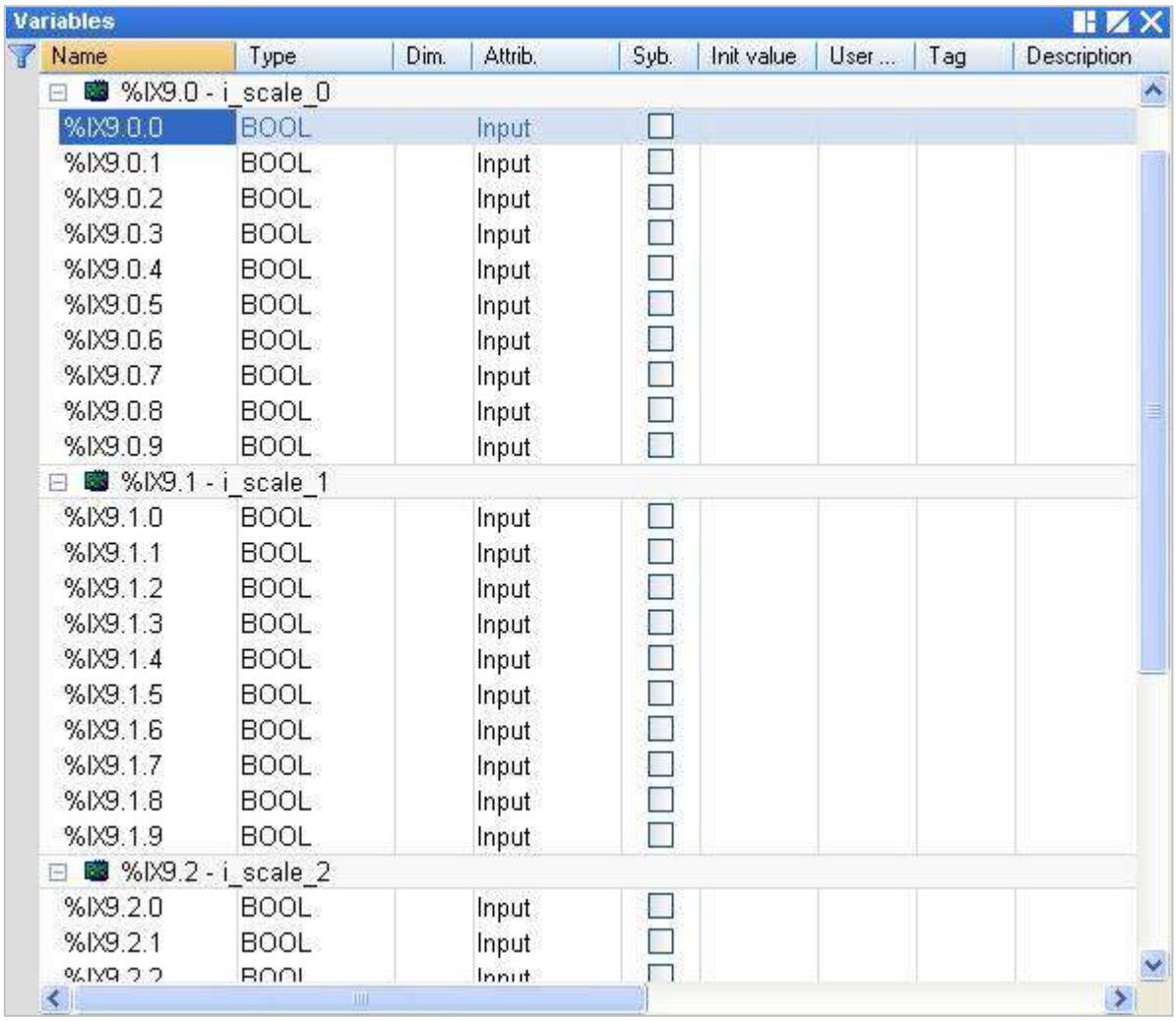
For example, if the AO board's value is -10 to +10 V and their respective engineering value is -50 to 1200, then set Ch_X0 as "-10.0", Ch_X1 as "+10.0", Ch_Y0 as "-50.0", Ch_Y1 as "+1200.0".



3. After linking the "i_scale" in the "I/O Boards" window, it will auto add 30 Boolean variables in the "Variables" window. When the Win-GRAF connects the PAC, it will display the state of each scaling function.

   True: scaling function is ok.

   FALSE: scaling function is not enabled or setting error.

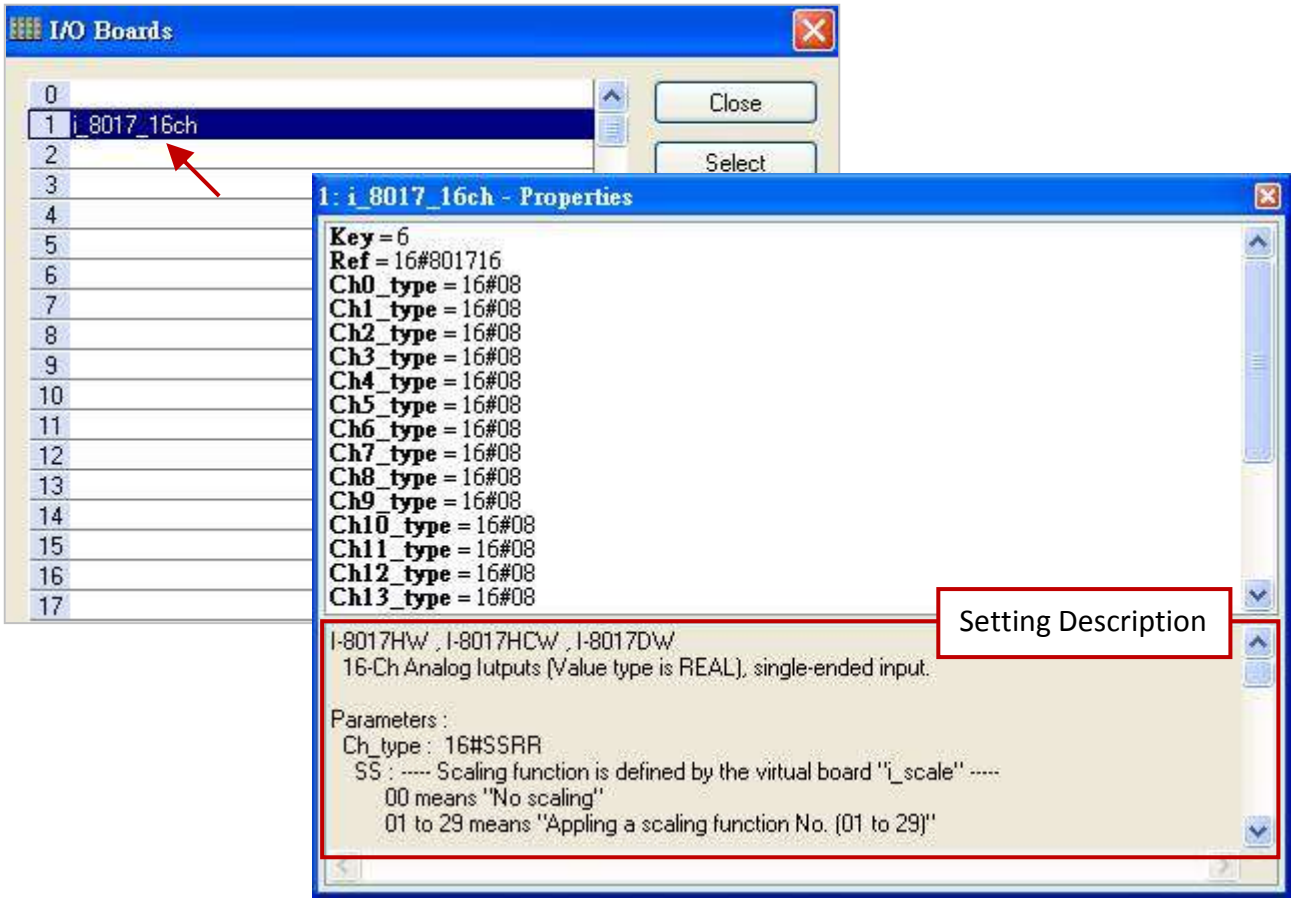


4.3 i_8017HW (8/16 channels AI)

The I-8017HW can be an 8-channel differential or a 16-channel single-ended analog input module (Data type: "REAL"). The following will describe a 16-channel module, you can see the Chapter4 (P4-1) to add this I/O board.

Note: Before using the I-8017HW I/O module, it requires to set the Differential or Single-ended Jumper in the hardware.

1. Mouse double click the "i_8017_16ch" to open the "Properties" window, and then to see the setting description.



Parameters:

Ch_type: 16#SSRR

SS : Scaling function is defined by the "i_scale" I/O board (see the Section 4.2).

00 means "No scaling".

01 to 29 means "Appling a scaling function No. (01 to 29)".

Setting "SS" as other value will use the default value 00.

RR : Range definition of signals.

05 means "physical input signal is -2.5 to +2.5 Volt".

06 means "physical input signal is -20 to +20 mA".

07 means "physical input signal is -1.25 to +1.25 Volt".

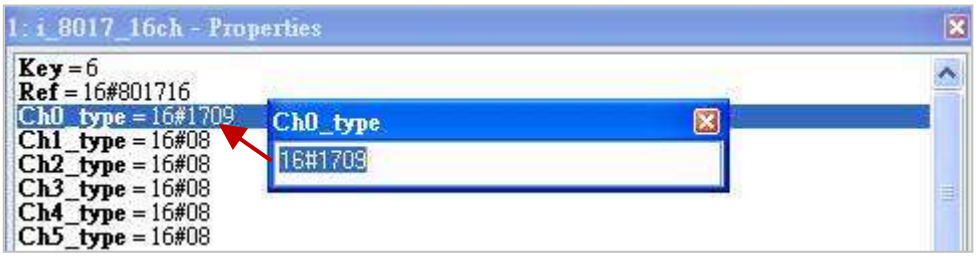
08 means "physical input signal is -10 to +10 Volt".

09 means "physical input signal is -5 to +5 Volt".

Setting "RR" as other value will use the default value 08.

2. Double click the item and fill in a value, then press the “Enter” key to complete the setting.

For example, 16#08 means the physical input signal is -10 to +10 V.
Channel value 5.67 means the input signal is 5.67 V.
For example, 16#209 means the physical input signal is -5 to +5 V with the scaling function 2.
Signal 5.67 V will be scaled to an engineering value by the scaling function 2.
For example, 16#1709 means the physical input signal is -5 to +5 V with the scaling function 17.
Signal 5.67 V will be scaled to an engineering value by the scaling function 17.



Noise_Filter_Max: The max. of the physical value to be considered as noise.
The filter will filter out the signal value beyond it, default setting is "9999.9".
For example, set as “7.9”, signal larger than 7.9 V (or 7.9 mA) will be filtered out.

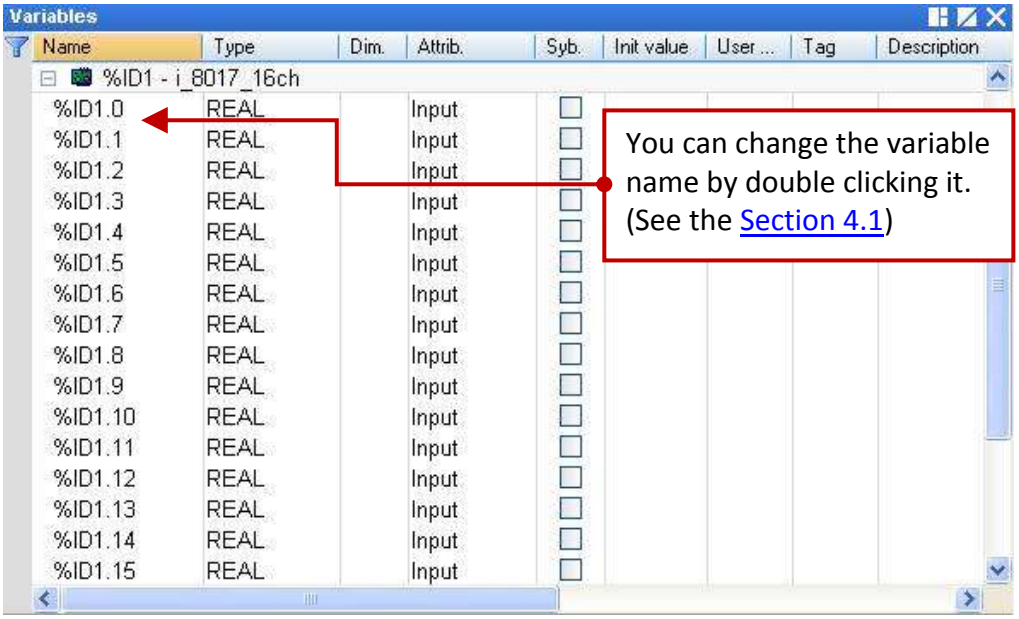
Noise_Filter_Min: The min. of the physical value to be considered as noise.
The filter will filter out the signal value beyond it, default setting is "-9999.9".
For example, set as “1.5”, signal smaller than 1.5 V (or 1.5 mA) will be filtered out.

Note:

If setting Noise_Filter_Min >= Noise_Filter_Max, filter is disabled.
If setting Noise_Filter_Min < -1000 and Noise_Filter_Max > 1000, the filter is disabled.

Sample_Number: The number of sampled data to be averaged as one data.
Default is “1” (range: 1 to 500). Set a bigger value will reduce the sampling rate, however the signal curve is smoother than setting a small value.

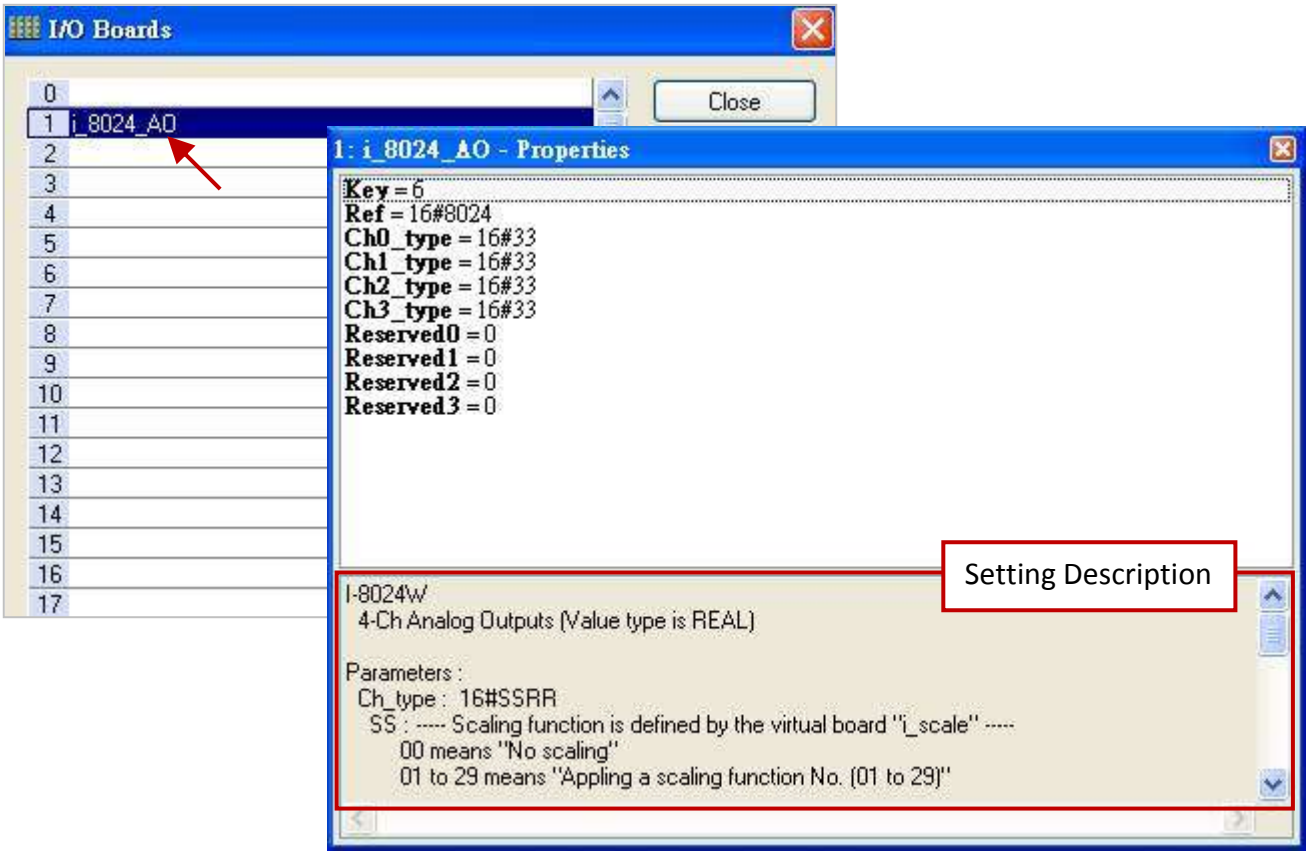
3. After linking the “i_8017_16ch” in the “I/O Boards” window, it will auto add 16 "REAL” input variables in the “Variables” window. These variables can be used in the program.



4.4 i_8024 (4-channel AO)

The I-8024W is a 4-channel analog output module (Data type: "REAL") that can be used to output +/- 10 V or 0 to +20 mA signal. See the Chapter4 (P4-1) to add this I/O board.

1. Mouse double-click the "i_8024_AO" to open the "Properties" window and then to see the setting description.



Parameters:

Ch_type: 16#SSRR

SS: Scaling function is defined by the "i_scale" I/O board (see the Section 4.2).

00 means "No scaling".

01 to 29 means "Appling a scaling function No. (01 to 29)".

Setting "SS" as other value will use the default value 00.

RR: Range definition of signals

30 means "physical output signal is 0 to 20 mA"

33 means "physical output signal is -10 to +10 Volt"

Setting "RR" as other value will use the default value 33.

2. Double click the item and fill in a value, then press the "Enter" key to complete the setting.

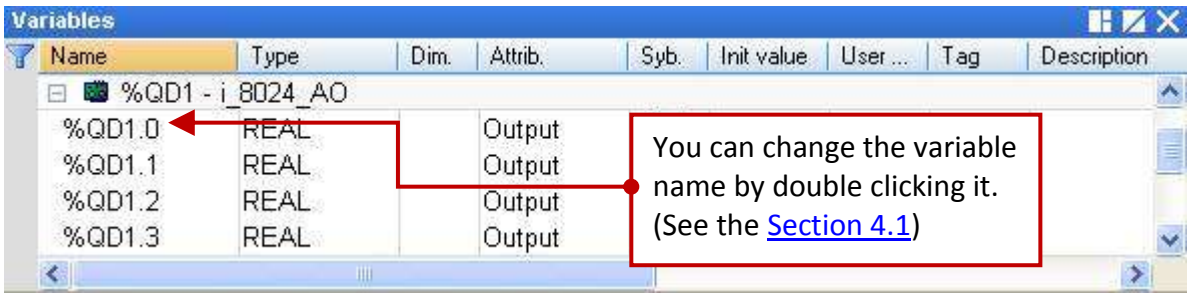
For example, 16#33 means the physical output signal is -10 to +10 V.
Channel value 5.67 is to output 5.67 V ; value -3.752 is to output -3.752 V.

For example, 16#133 means the physical output signal is -10 to +10 V with the scaling function 1.
Channel value is a user-defined engineering value will be converted first by the scaling function 1 (i.e., "Ch01") before output it as -10 to +10 V.

For example, 16#30 means the physical output signal is 0 to 20 mA.
Channel value 12.5 is to output 12.5 mA ; value 6.27 is to output 6.27 mA.

For example, 16#1730 means the physical output signal is 0 to 20 mA with the scaling function 17.
Channel value is a user-defined engineering value will be converted first by the scaling function 17 (i.e., "Ch17") before output it as 0 to 20 mA.

3. After linking the “i_8024_AO” in the “I/O Boards” window, it will auto add 4 "REAL” Output variables in the “Variables” window. These variables can be used in the program.



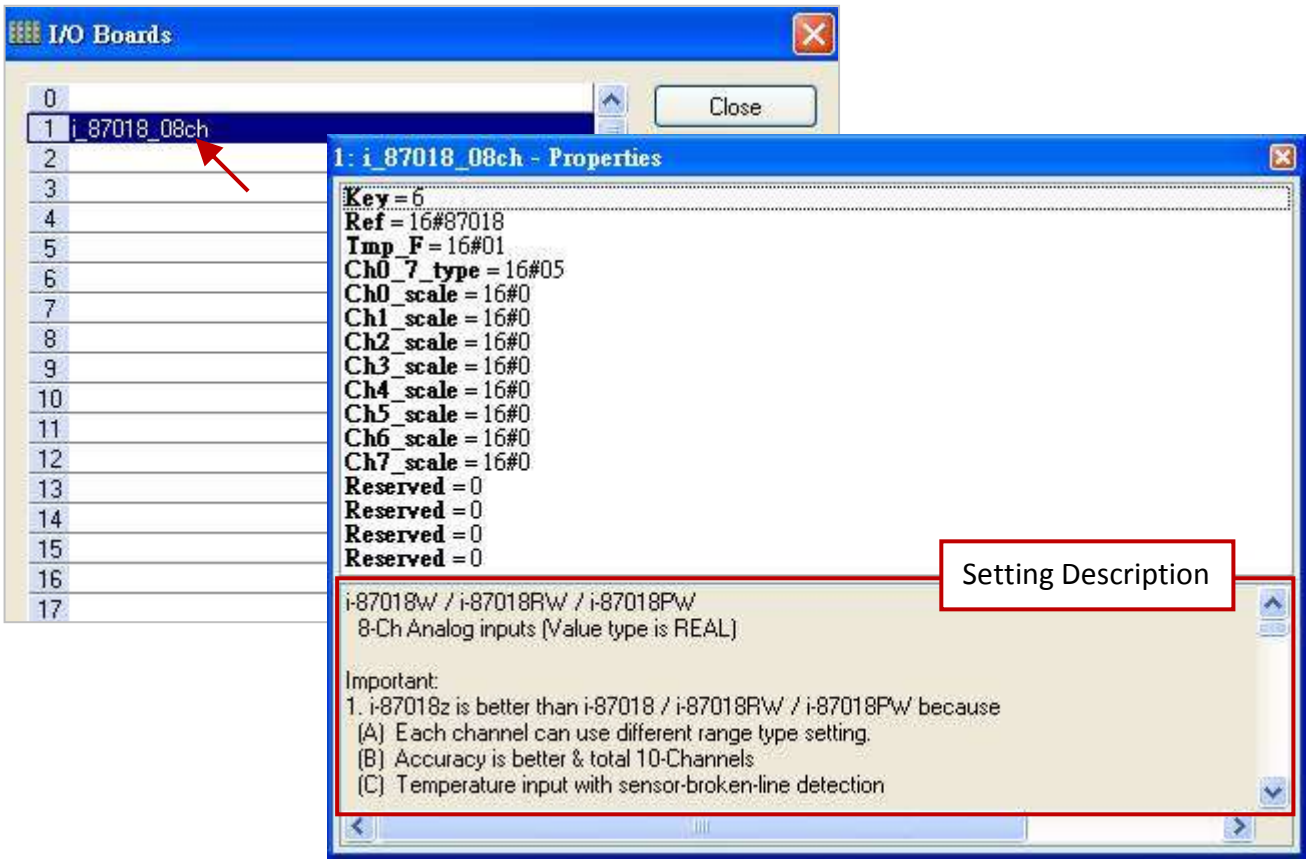
4.5 i_87018W (8-channel AI)

The I-87018W is an 8-channel analog input module (Data type: "REAL") that provides thermocouple input, current input (-20 mA to +20 mA) and voltage input (+/- 15 mV, +/- 50 mV, +/- 100 mV, +/- 500 mV, +/- 1 V, +/- 2.5 V). See the Chapter4 (P4-1) to add this I/O board.

Important Notice:

- I-87018ZW is better than I-87018W / I-87018RW / I-87018PW because
  (A) Each channel can use different range type setting.
  (B) Accuracy is better and total 10-Channels.
  (C) Temperature input with sensor-broken-line detection.
  Please visit http://www.icpdas.com/products/Remote_IO/i-87k/i-87018z.htm
- I-87018W does not support sensor-broken line function.

1. Mouse double-click the "i_8018_08ch" to open the "Properties" window, and then to see the setting description.



Parameters:

Tmp_F: 16#FF

FF: Temperature format, It only apply to the channel type is Thermocouple.

01 means the unit of the input value is Degree Celsius

02 means the unit of the input value is Degree Fahrenheit

Ch0_7_type: 16#RR

RR: Range definition of signals.

Normal Range: (For mA or Volt)

Type Code	Physical Input Signal
00	-0.015 to +0.015 V
01	-0.05 to +0.05 V
02	-0.1 to +0.1 V
03	-0.5 to +0.5 V
04	-1 to +1 V
05	-2.5 to +2.5 V
06	20 to +20 mA

When I-87018 and I-87018R are connected to a current source and set to “06” type code, an optional external 125 Ohm resistor is required.

Thermocouple Range: (For temperature)

Type Code	Type	Physical Input Signal
0E	J	-210 to +760 °C
0F	K	-270 to +1372 °C
10	T	-270 to +400 °C
11	E	-270 to +1000 °C
12	R	0 to +1768 °C
13	S	0 to +1768 °C
14	B	0 to +1820 °C
15	N	0 to +2320 °C
17	L	-200 to +800 °C
18	M	-200 to +100 °C
19	$L_{DIN43710}$	-200 to +900 °C

Setting RR as other value will use the default value “05”

2. Double click the item and fill in a value, then press the “Enter” key to complete the setting.

   For example, 16#05 means the physical input signal is -2.5 to +2.5 V.
   Channel value 1.28 means the input signal is 1.28 V.
   Channel value -0.752 means the input signal is -0.752 V.

   For example, If "Ch0_7_type" set as 16#0F and "Tmp_F" set as 16#01 means the physical input signal is -270 to +1372 degree Celsius.
   Channel value 25.75 means 25.75 degree Celsius.

   For example, If "Ch0_7_type" set as 16#10 and "Tmp_F" set as 16#01 means the physical input signal is -454 to +752 degree Fahrenheit.
   Channel value 25.75 means 25.75 degree Fahrenheit.

Note: If using a temperature module with a broken-line detection function (e.g., I-87018ZW), and the temperature value is greater than "9000.0", it means,

1. The temperature sensor may be broken-line.
2. The temperature sensor may be damaged.
3. The DCON module is not configured well to fit the connected temperature sensors.
4. The ohm measured by the connected sensor is not correct.

Ch0_scale to Ch7_scale: 16#SS

SS: Scaling function is defined by the "i_scale" I/O board (refer the Section 4.2).

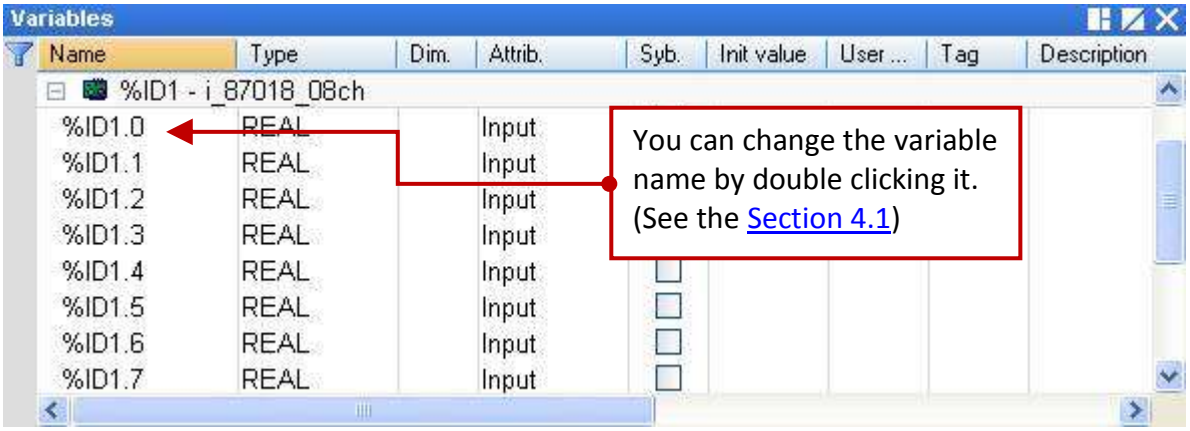
00 means "No scaling".

01 to 29 means "Appling a scaling function No. (01 to 29)"

Setting SS as other value will use the default value 00.

For example, 16#17 means the physical input signal is converted with the scaling function 17.

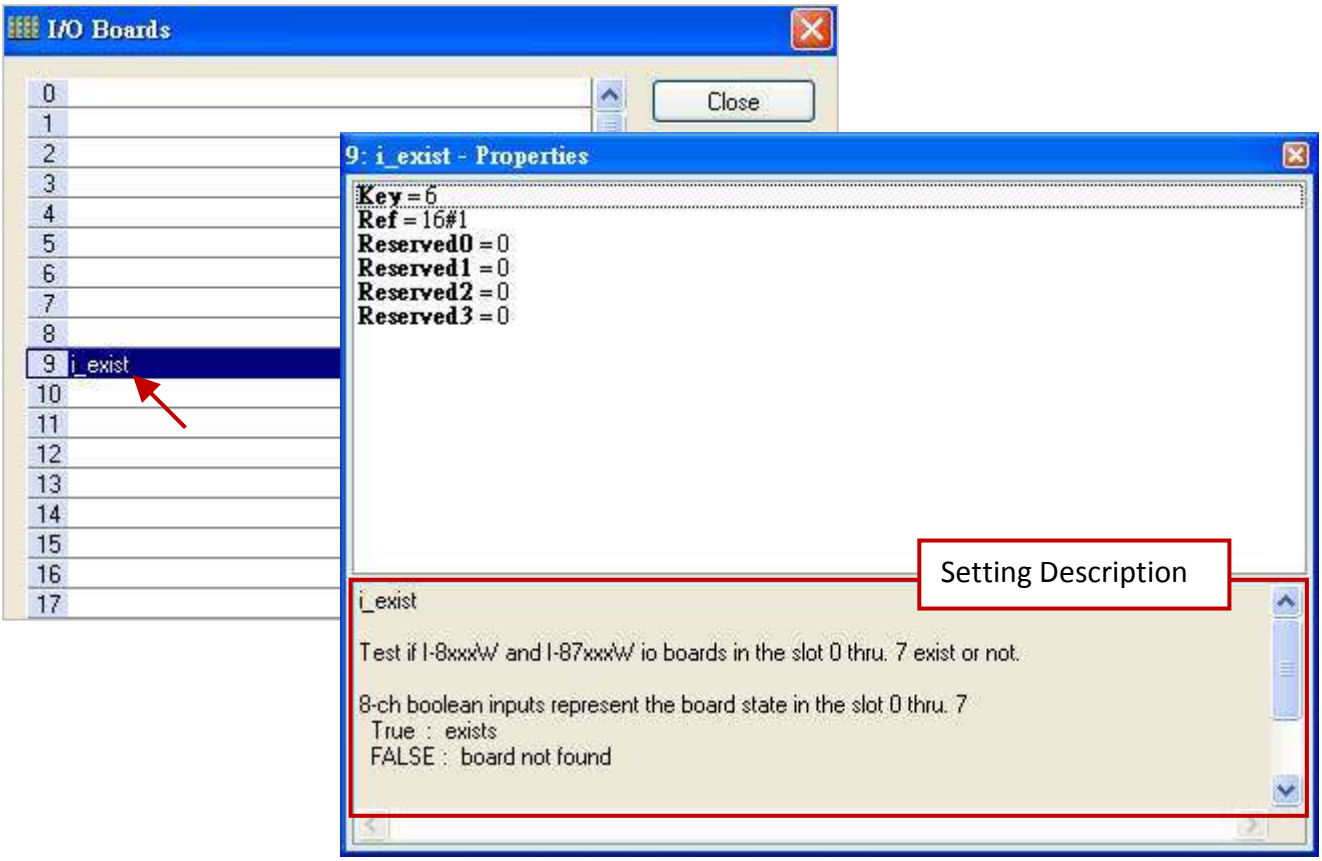
3. After linking the “i_87018_08ch” in the “I/O Boards” window, it will auto add 8 "REAL” input variables in the “Variables” window that are available for programming.



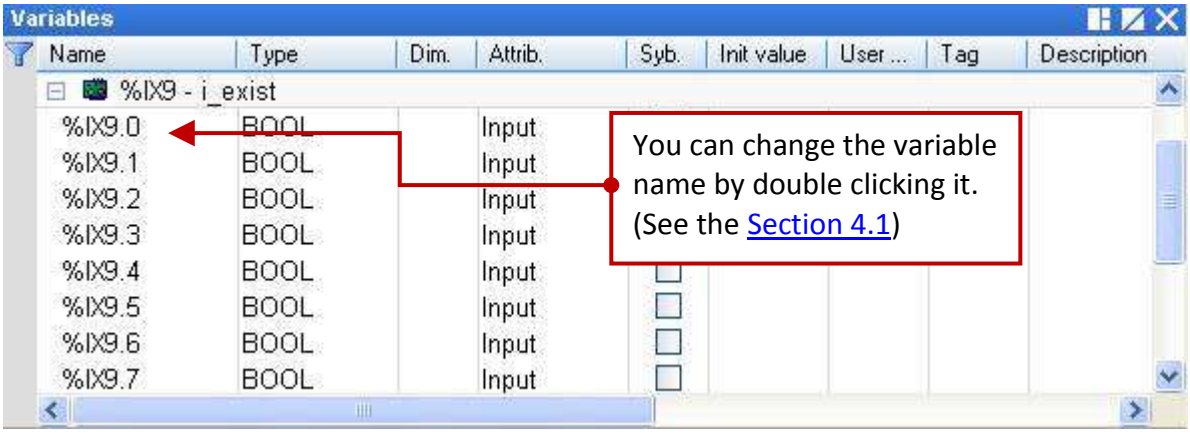
4.6 i_exist (Test if the I/O module exists?)

The “i_exist” is used to check if the I-8K and I-87K series I/O modules exist in the PAC‘s slot 0 to 7.
See the Chapter4 (P4-1) to add this I/O board.

1. Mouse double-click the "i_exist" to open the “Properties” window, and then to see the setting description.
   Note: Using the Slot 8 or above No. because the slot 0 to slot 7 are reserved for the real I/O module.



2. After linking the “i_exist” in the “I/O Boards” window, it will auto add 8 "BOOL” input variables in the “Variables” window and display the state of the I/O module from slot 0 to slot 7 when connecting the Win-GRAF PAC.
   "TRUE" means the I/O module exists.
   "FALSE" means cannot find this I/O module.

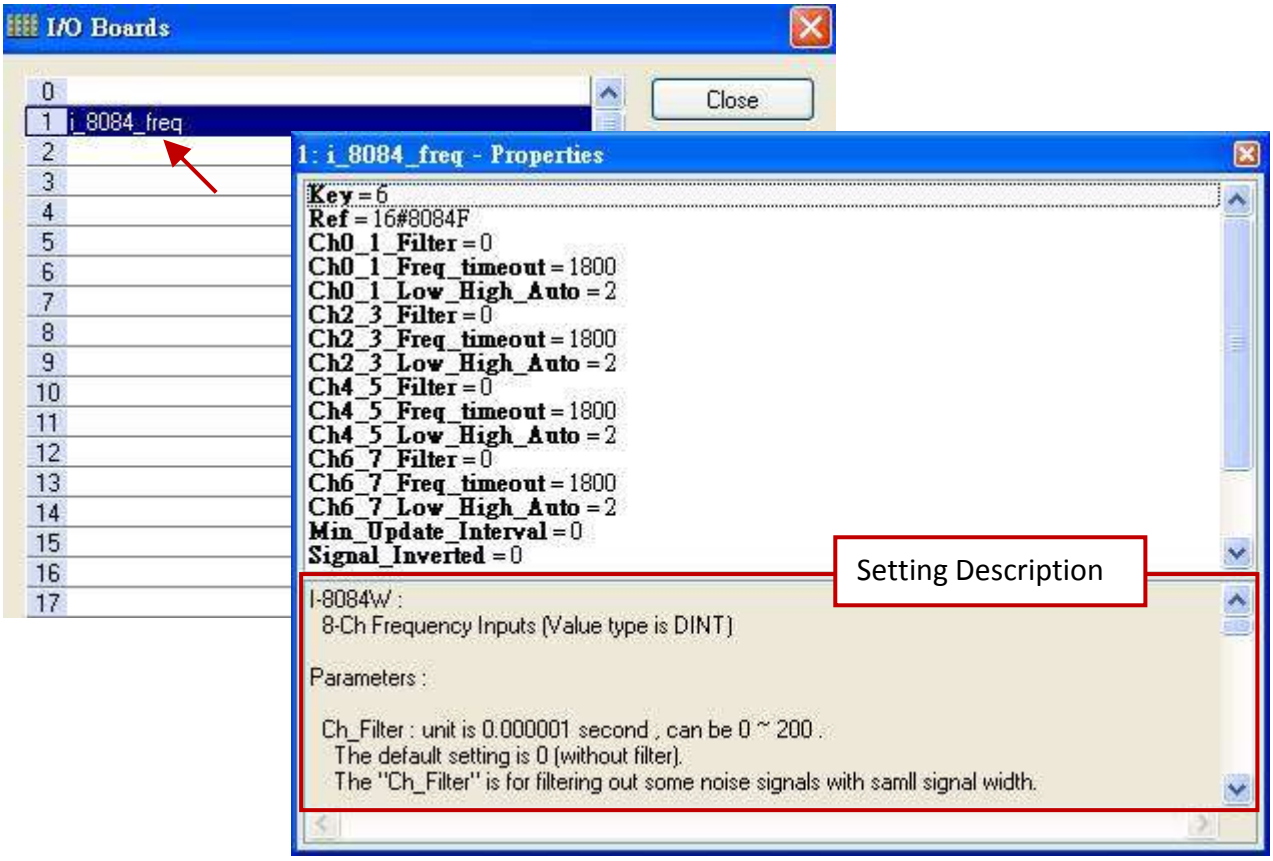


4.7 i_8084 (Frequency, UP/Down Counter, UP Counter)

The I-8084W is a 4/8-channel high speed Frequency/Counter (Data type: “DINT”) that can be used to measure frequency or as a UP/Down Counter or as a UP Counter. The following will describe these three modes. See the Chapter4 (P4-1) to add this I/O board.

4.7.1 i_8084_freq (8-channel Frequency)

1. Mouse double-click the "i_8084_freq" to open the “Properties” window, and then to see the setting description.



Parameters:

Ch_Filter: The unit is 0.000001 second (μs), the value can be 0 to 200.
The default setting is 0 (without filter). The "Ch_Filter" is for filtering out some noise signals with smaller signal width. (Recommend 0: if there is no noise consideration or need a real-time measurement.) The following setting is recommended:

Max Input Signal (Hz)	Recommend Filter Value
1K	200
2K	100
5K	40
10K	20
20K	10
100K	2
450K	1
450K	0 (without filter)

Ch_Freq_Timeout:

The unit is 0.001 second (ms), the value can be 20 to 1800. Set as other value will use the default value 1800. If there is no signal wave input to the I-8084W in the "Ch_Freq_Timeout" interval, the frequency value of the related channel will be assigned as 0.

For example, if set it as 100 ms and the input is 500 Hz (that means, one signal wave takes about 2 ms to happen), the frequency is updated normally. When the input frequency drop to 9 Hz (that means, one signal wave take about 111 ms to happen), this "111" exceeds the setting "100" ms (Freq_Timeout). So the frequency value will be assigned as 0 because there is no signal wave coming in this 100 ms interval.

When setting as 20 ms, the frequency value below 50 Hz is not detectable (become 0).
When setting as 100 ms, the frequency value below 10 Hz is not detectable (become 0).
When setting as 1800 ms, the frequency value of 0 Hz , 1 Hz to 450 KHz is detectable.

Ch_Low_High_Auto: (recommend setting as “2: Auto”.)

0 means a Low frequency mode.
1 means a High frequency mode.
2 means Auto switching between Low and High frequency mode.
Set as other value will use the default value “2: Auto”.

Mode 2 will auto change the frequency mode. It will auto change to High mode when the input frequency is larger than 3500 Hz, while auto change to Low mode if input frequency is less than 1000 Hz.

DO NOT set as 1 (High frequency mode) if the input signal is normally less than 1000 Hz, or the frequency value will be incorrect frequently. Recommended don't set as 0 (Low frequency mode) if the input signal is normally larger than 3500 Hz.

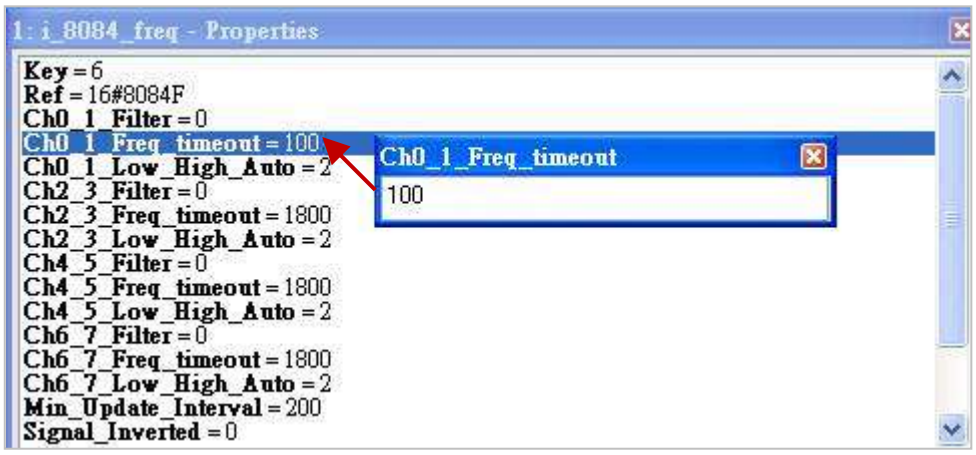
Min_Update_Interval:

The unit is 0.001 second (ms), the value can be 0, or 20 to 1000.
Default value 0 means "Update frequency every PAC cycle".
Other means "Update frequency when each Interval time reached".
The frequency update time also depends on the Win-GRAF PAC cycle time. If the PAC cycle time is big, for example 200 ms, then the real frequency update time will become 200 ms when setting the "Min_Update_Interval" less than 200. Setting bigger "Min_Update_Interval" will get smooth frequency curve value, however the frequency value is updated slowly.

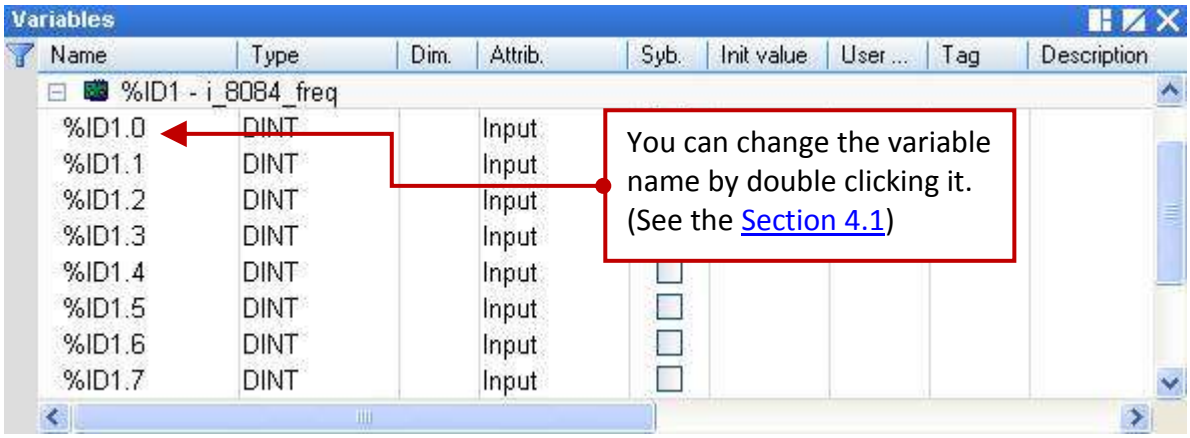
Signal_Inverted:

0 : input signal is normal (no inverted).
1 : input signal is inverted (means voltage HIGH will be processed as LOW, and voltage LOW will be processed as HIGH).

2. Double click the item and fill in a value, then press the “Enter” key to complete the setting.



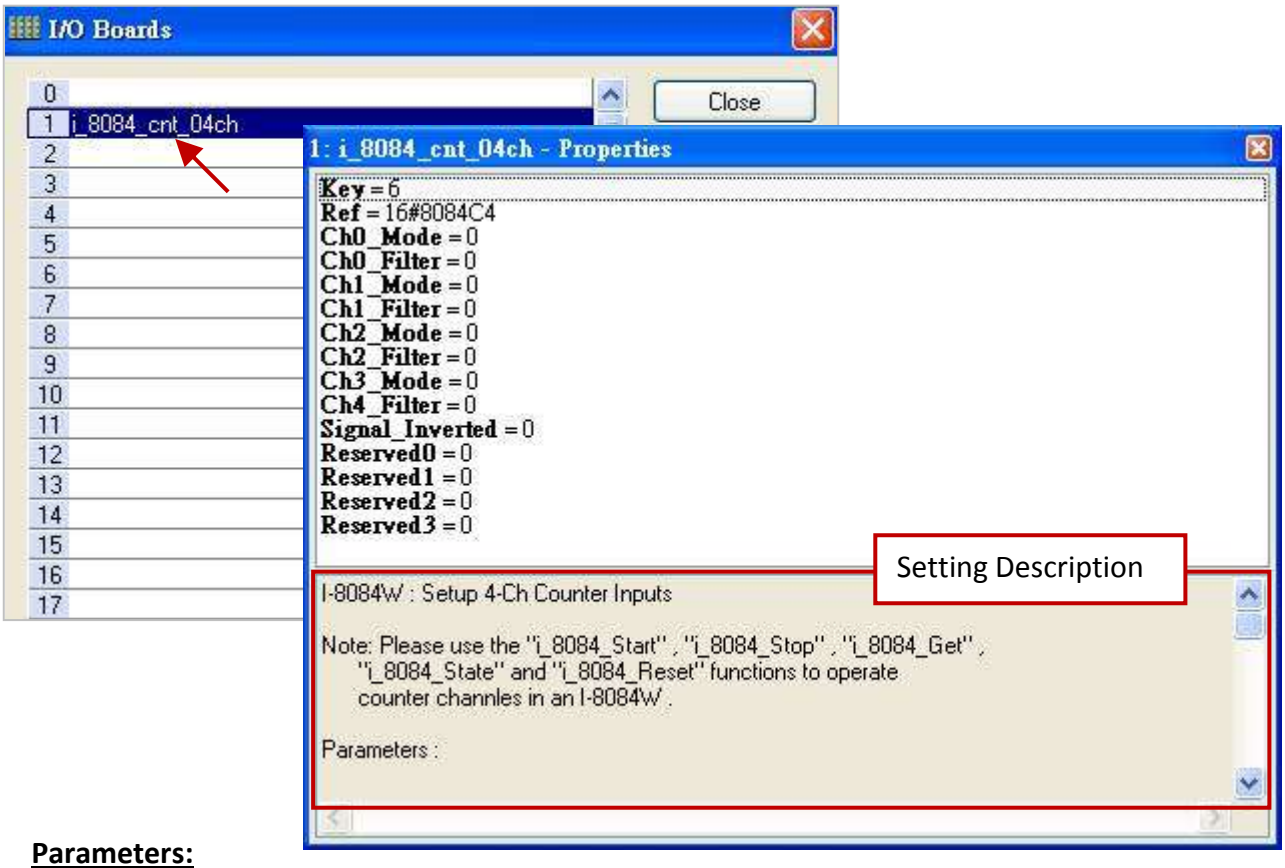
3. After linking the “i_8084_freq” in the “I/O Boards” window, it will auto add 8 “DINT” input variables in the “Variables” window. When the Win-GRAF connects the PAC, it will display the frequency value for each channel.



4.7.2 i_8084_cnt_ch04 (4-channel UP/Down Counter)

Note: Using the "COUNTER_START", "COUNTER_STOP", "COUNTER _GET", "COUNTER_STATE" and "COUNTER_RESET" functions in the Win-GRAF Workbench to operate counter channels in an I-8084W.

1. Mouse double-click the "i_8084_cnt_ch04" to open the "Properties" window, and then to see the setting description.



Parameters:

Ch_Mode: Input mode, can be 0 , 1 and 4. Set other value will use 0.

0: Pulse/DIR mode.

1: UP/DOWN mode.

4: A/B phase (Quard.) mode.

Ch_Filter: The unit is 0.000001 second (μs), the value can be 0 to 200.
The default setting is 0 (without filter). The "Ch_Filter" is for filtering out some noise signals with smaller signal width. (Recommend 0: if there is no noise consideration or need a real-time measurement.) The following setting is recommended:

Max Input Signal (Hz)	Recommend Filter Value
1K	200
2K	100
5K	40
10K	20
20K	10
100K	2
450K	1
450K	0 (without filter)

Signal_Inverted:

0: Input signal is normal (no inverted).

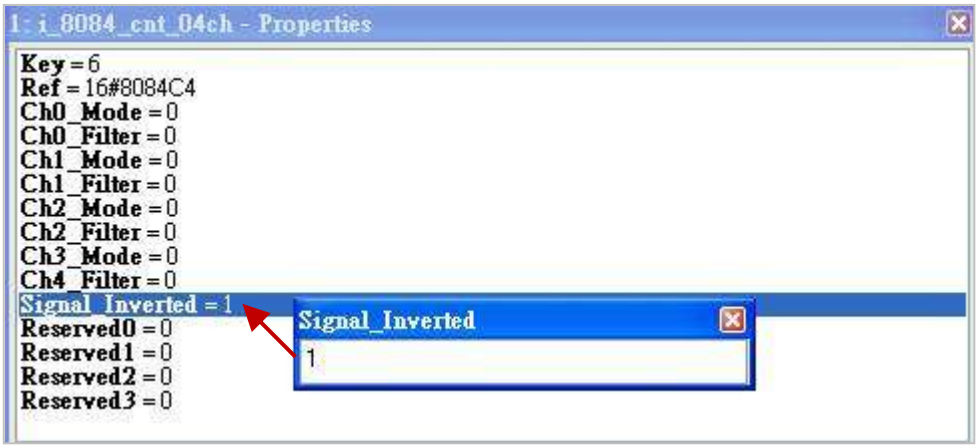
1: Input signal is inverted (means voltage HIGH will be processed as LOW, and voltage LOW will be processed as HIGH).

For example:

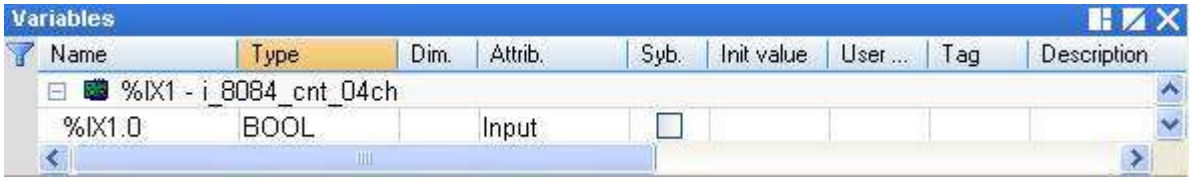
If setting "Signal_Inverted" as 0 (no inverted) and Ch_Mode is 0 (Pulse/DIR), the counter value will count up if "DIR" signal is High.

If setting "Signal_Inverted" as 1 (inverted) and Ch_Mode is 0 (Pulse/DIR), the counter value will count down if "DIR" signal is High.

2. Double click the item and fill in a value, then press the “Enter” key to complete the setting.



3. After linking the “i_8084_cnt_ch04” in the “I/O Boards” window, it will auto add one “BOOL” Input variable (no meaning, always “FALSE”) in the “Variables” window.

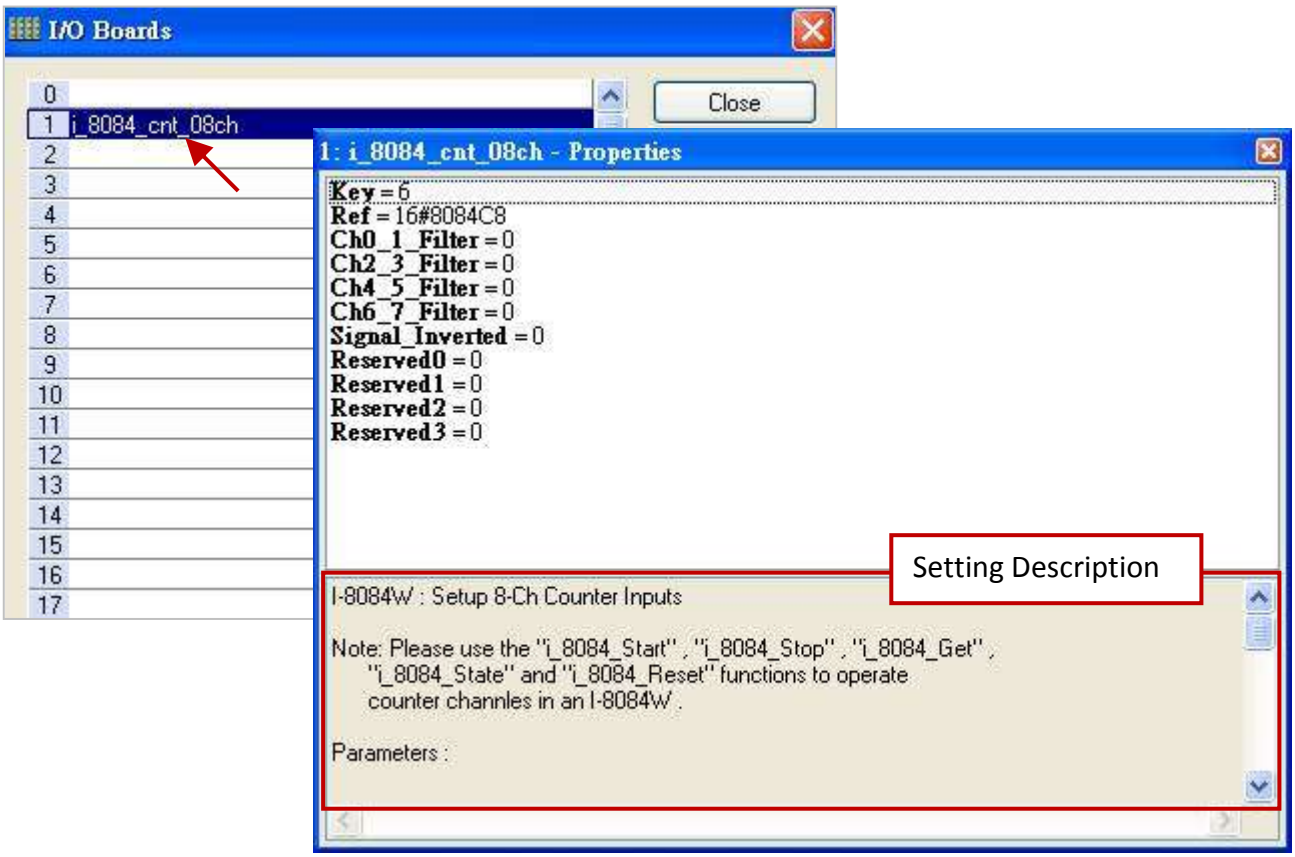


4. After linking the “I/O board”, refer the Section 4.9 to use "COUNTER_START", "COUNTER_STOP", "COUNTER _GET", "COUNTER_STATE" and “COUNTER_RESET” functions in the LD or ST program to operate the Counter channel of the I-8084W.

4.7.3 i_8084_cnt_ch08 (8-channel UP Counter)

Note: Using the "COUNTER_START", "COUNTER_STOP", "COUNTER _GET", "COUNTER_STATE" and "COUNTER_RESET" functions in the Win-GRAF Workbench to operate counter channels in an I-8084W.

1. Mouse double-click the "i_8084_cnt_ch08" to open the "Properties" window, and then to see the setting description.



Parameters:

Ch_Filter: The unit is 0.000001 second (μs), the value can be 0 to 200.
The default setting is 0 (without filter). The "Ch_Filter" is for filtering out some noise signals with smaller signal width. (Recommend 0: if there is no noise consideration or need a real-time measurement.) The following setting is recommended:

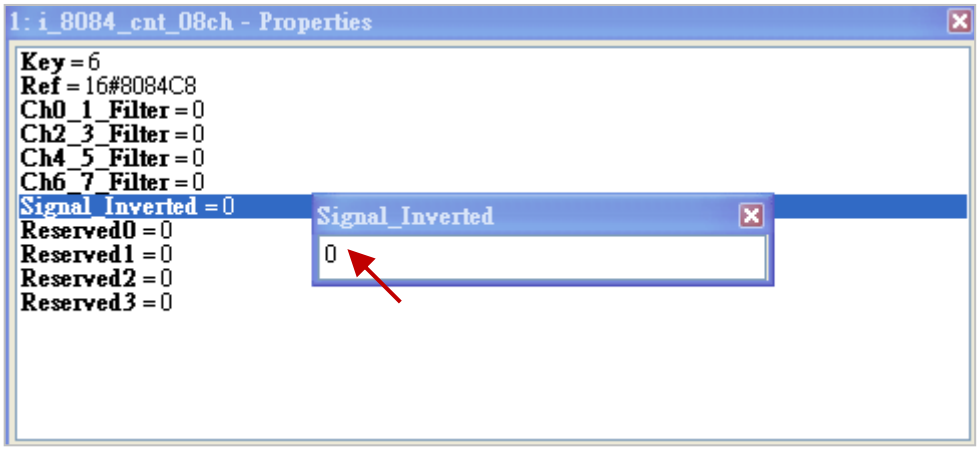
Max Input Signal (Hz)	Recommend Filter Value
1K	200
2K	100
5K	40
10K	20
20K	10
100K	2
450K	1
450K	0 (without filter)

Signal_Inverted:

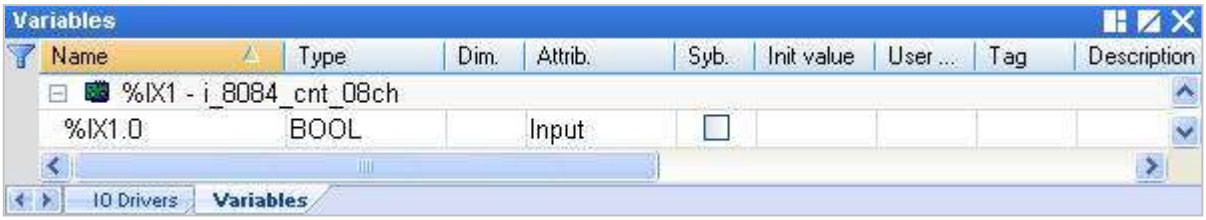
0: Input signal is normal (no inverted)

1: Input signal is inverted (means voltage HIGH will be processed as LOW, and voltage LOW will be processed as HIGH).

2. Double click the item and fill in a value, then press the “Enter” key to complete the setting.



3. After linking the “i_8084_cnt_ch08” in the “I/O Boards” window, it will auto add one “BOOL” Input variable (no meaning, always “FALSE”) in the “Variables” window.

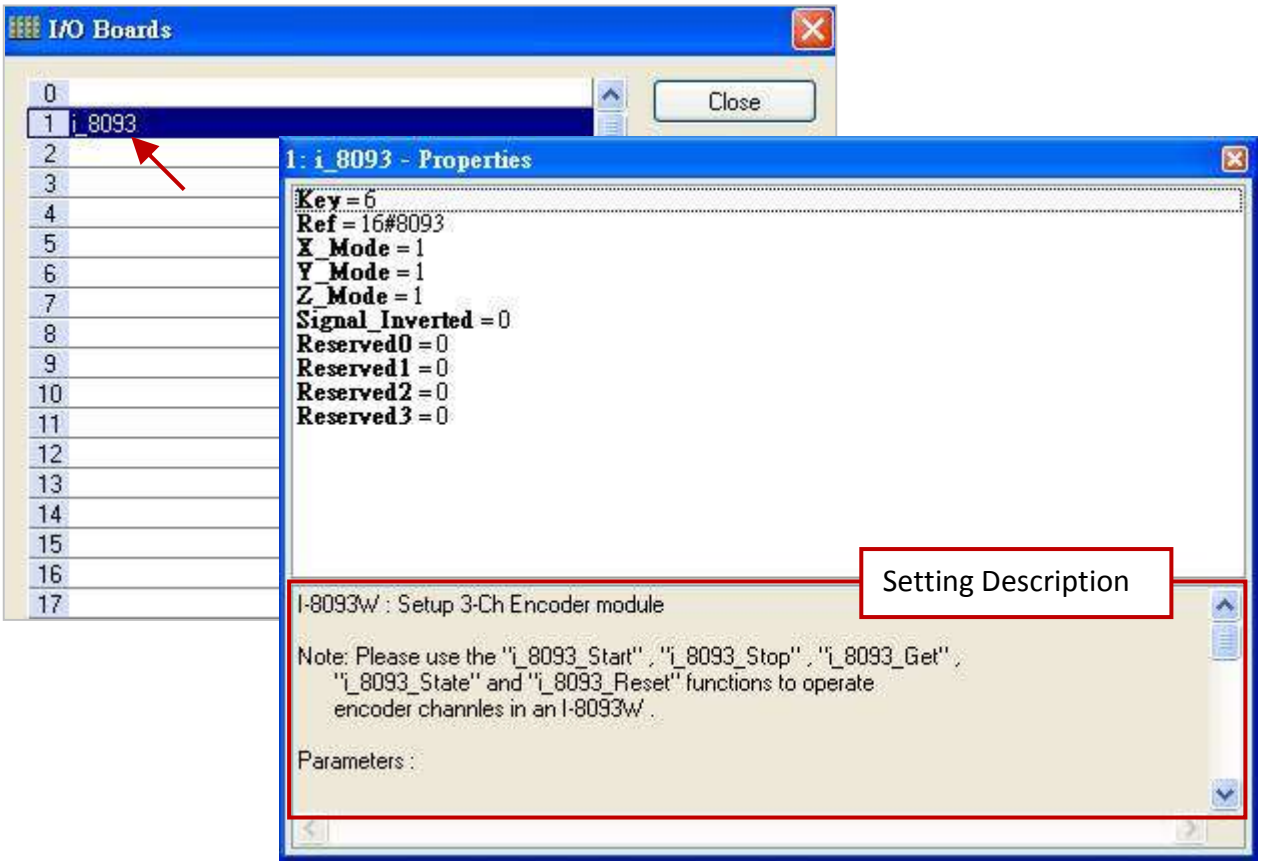


4. After linking the “I/O board”, refer the Section 4.9 to use "COUNTER_START", "COUNTER_STOP", "COUNTER _GET", "COUNTER_STATE" and “COUNTER_RESET” functions in the LD or ST program to operate the Counter channel of the I-8084W.

4.8 i_8093 (3-axis High Speed Encoder Module)

The I-8093W is a 3-axis high speed encoder module that can be independently configured as one of the Quadrant, Pulse/Direction or CW/CCW input mode for each channel. If not familiar with the way to add the this I/O board, see the Chapter4 (P4-1).

1. Mouse double-click the "i_8093" to open the "Properties" window, and then to see the setting description.



Parameters:

X_Mode, Y_Mode, Z_Mode:

The input mode of X, Y, Z axis, can be 1, 2 and 3. Set other value will use 1.

1: CW/CCW counting mode.
2: Pulse/Directioncounting mode.
3: A/B phase (quadrant) counting mode.

Signal_Inverted:

0: Input signal is normal (no inverted)
1: Input signal is inverted (means voltage HIGH will be processed as LOW, and voltage LOW will be processed as HIGH).

For example:

If setting "Signal_Inverted" as 0 (no inverted) and X_Mode is 2 (Pulse/Direction), the encoder value will increase if "Direxion" signal is High.
If setting "Signal_Inverted" as 1 (inverted) and X_Mode is 2 (Pulse/Direction), the encoder value will decrease if "Direxion" signal is High.

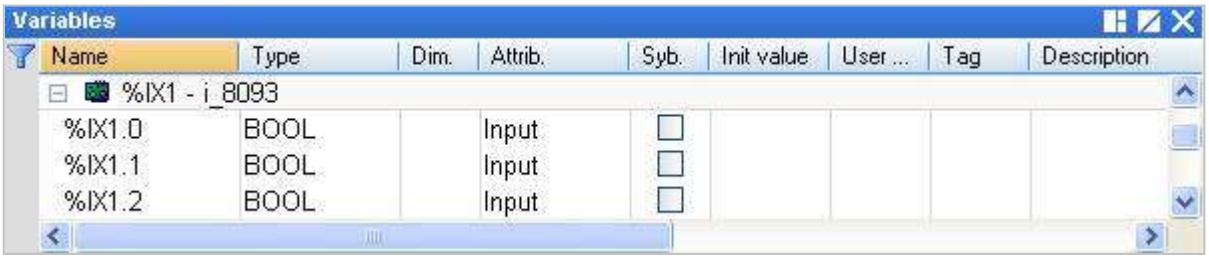
2. Double click the item and fill in a value, then press the “Enter” key to complete the setting.

3. After linking the “i_8093” in the “I/O Boards” window, it will auto add 3 “BOOL” input variables in the “Variables” window that are available for programming.

   Ch0: Z-index of X axis.

   Ch1: Z-index of Y axis.

   Ch2: Z-index of Z axis.



4. After linking the “I/O Boards”, refer the Section 4.9 to use "COUNTER_START", "COUNTER_STOP", "COUNTER _GET", "COUNTER_STATE" and “COUNTER_RESET” functions in the LD or ST program to operate the Encoder channel of the I-8093W.

4.9 Using the Count Function for I-8084W, I-8093W, I-87082W, I-87084W, I-7083 and I-7080 Modules

This section lists the way to use the "COUNTER_START", "COUNTER_STOP", "COUNTER _GET", "COUNTER_STATE" and "COUNTER_RESET" functions in the LD or ST program to operate the Counter and Encoder modules. If not familiar with the way to create a program or a function block, see the Section 2.3.3.

Note:

1. **In the following content, we use I-8084W and I-8093W modules as examples.**
2. **Before using these function blocks, first go to Section 4.7.2 (UP/Down Counter), Section 4.7.3 (UP Counter) and Section 4.8 (Encoder) to link I/O Boards.**

4.9.1 COUNTER_START

For example: Using the I-8084W module in the PAC's slot2 and start counting the channel 5.

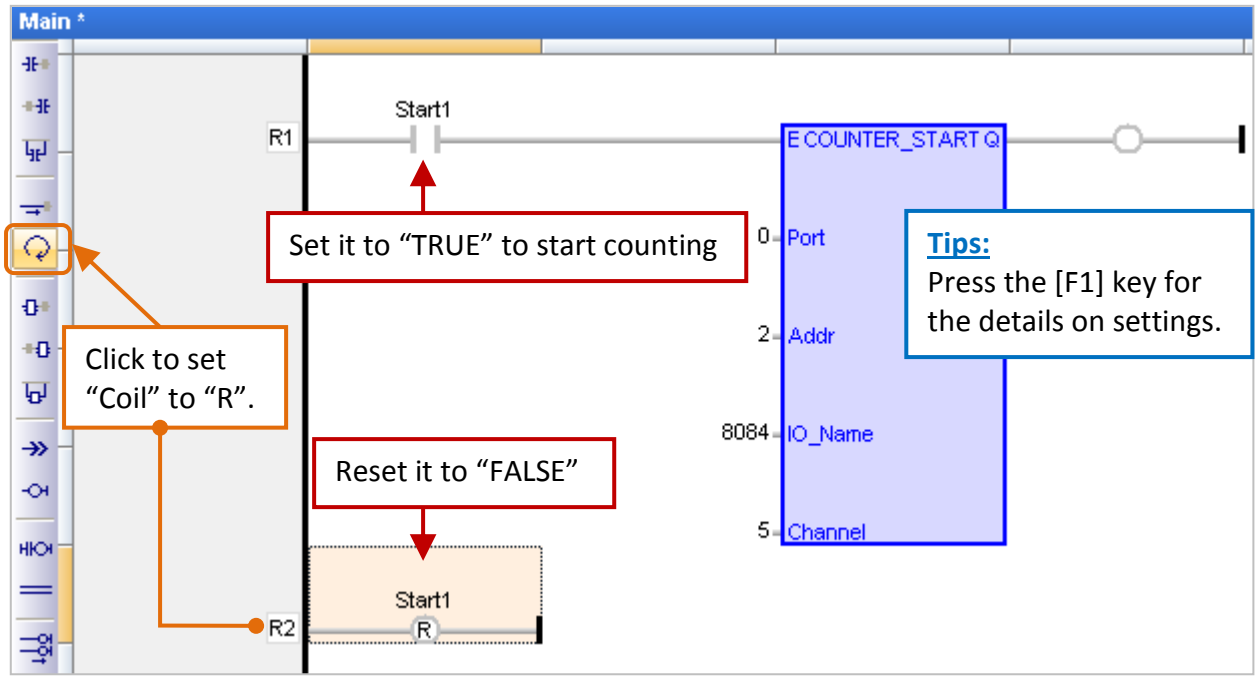
ST program:

```
IF   Start1 = TRUE   THEN
  Start1  :=  FALSE ;
  TMP_BOOL  :=  Counter_Start (0, 2, 8084, 5) ;
END_IF ;
```

Note:
First, add two BOOL variables ("Start1", "TMP_BOOL") in the Variables Area.

LD program:

("Start1": boolean, set it to "TRUE" to start counting and then reset "Start1" to "FALSE".)



Port: (Data type: “DINT”)
For a module in the slot 0 to 7 of the PAC, set it as “0”. For a DCON module connected to a serial COM port, can be “1 to 37” (depends on PAC, means COM1 to COM37).

Addr: (Data type: “DINT”)
For a module in the slot 0 to 7 of the PAC, set it as the slot number (0 to 7).
For a module connected to a serial COM port, set it as the Net-ID address of the module (1 to 255).

IO_Name: (Data type: “DINT”)
The name of relative Counter/Encoder module, it can be set to "8084”, "8093", "87084", "87082", "7083" and "7080".

Channel: (Data type: “DINT”)
The channel No. of the Counter/Encoder module, it can be set to "0", "1", and so on, depends on the module. For example, when using the I-8093W module, "0" means the X-axis, "1" means the Y-axis and "2" means the Z-axis.

Q: (Data type: “BOOL”)
“TRUE”: OK ; “FALSE”: Error.

4.9.2 COUNTER_STOP

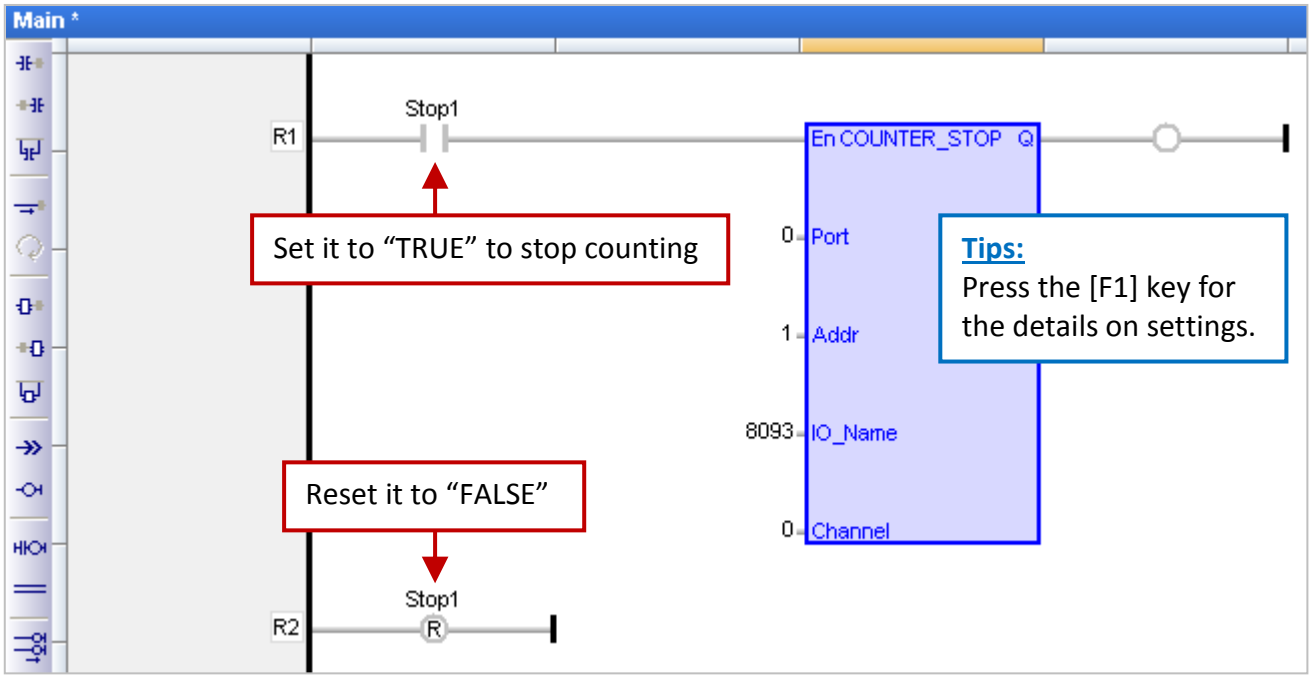
For example: Using the I-8093W module in the PAC's slot1 and stop counting the X-axis.

ST program:

```
IF   Stop1 = TRUE   THEN
  Stop1  :=  FALSE ;
  TMP_BOOL  :=  Counter_Stop (0, 1, 8093, 0) ;
END_IF ;
```

Note:
First, add two BOOL variables ("Stop1", "TMP_BOOL") in the Variables Area.

LD program:
("Stop1": Boolean, set it to "TRUE" to start counting and then reset "Stop1" to "FALSE".)



Port: (Data type: "DINT")
For a module in the slot 0 to 7 of the PAC, set it as "0". For a DCON module connected to a serial COM port, can be "1 to 37" (depends on PAC, means COM1 to COM37).

Addr: (Data type: "DINT")
For a module in the slot 0 to 7 of the PAC, set it as the slot number (0 to 7).
For a module connected to a serial COM port, set it as the Net-ID address of the module (1 to 255).

IO_Name: (Data type: "DINT")
The name of relative Counter/Encoder module, it can be set to "8084", "8093", "87084", "87082", "7083" and "7080".

Channel: (Data type: "DINT")
The channel No. of the Counter/Encoder module, it can be set to "0", "1", and so on, depends on the module. For example, when using the I-8093W module, "0" means the X-axis, "1" means the Y-axis and "2" means the Z-axis.

Q: (Data type: "BOOL")
"TRUE": OK ; "FALSE": Error.

4.9.3 COUNTER_GET

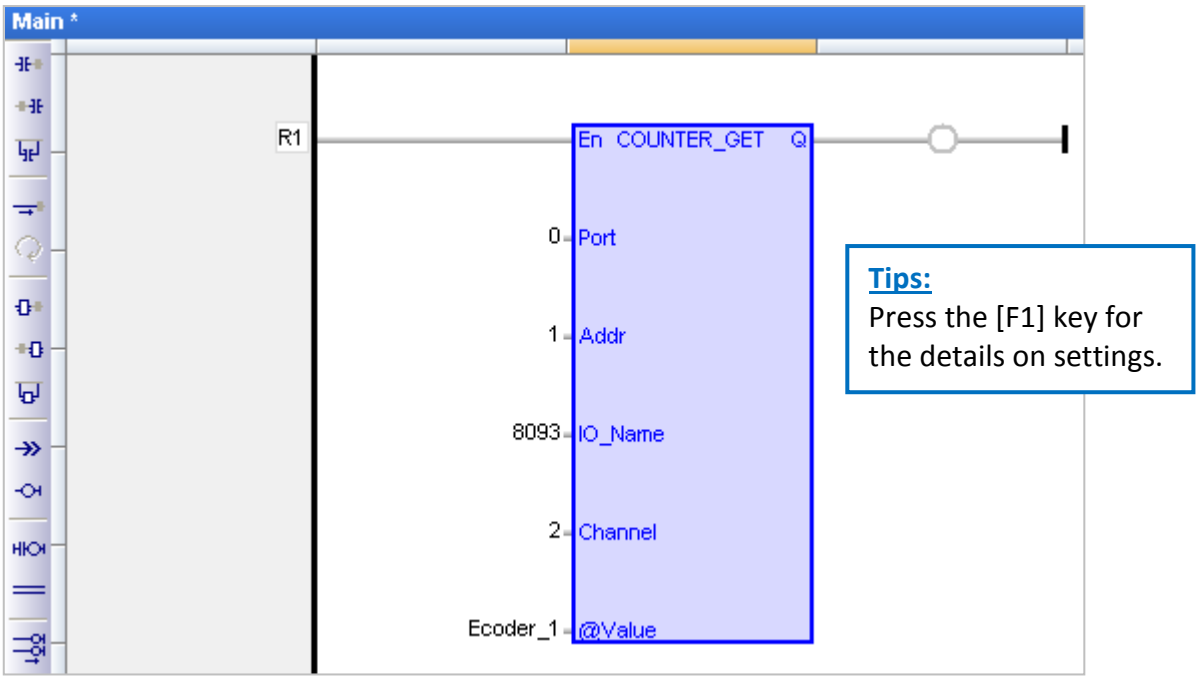
For example: Using the I-8093W module in the PAC's slot1 and get the Encoder value of the Z-axis.

ST program:

```
TMP_BOOL := Counter_Get (0, 1, 8093, 2, Encoder_1) ;
```

Note: First, add variables in the Variables Area (see section 2.2.1).
"TMP_BOOL" (BOOL).
"Encoder_1" (DINT).

LD program:



Port: (Data type: "DINT")
For a module in the slot 0 to 7 of the PAC, set it as "0". For a DCON module connected to a serial COM port, can be "1 to 37" (depends on PAC, means COM1 to COM37).

Addr: (Data type: "DINT")
For a module in the slot 0 to 7 of the PAC, set it as the slot number (0 to 7).
For a module connected to a serial COM port, set it as the Net-ID address of the module (1 to 255).

IO_Name: (Data type: "DINT")
The name of relative Counter/Encoder module, it can be set to "8084", "8093", "87084", "87082", "7083" and "7080".

Channel: (Data type: "DINT")
The channel No. of the Counter/Encoder module, it can be set to "0", "1", and so on, depends on the module. For example, when using the I-8093W module, "0" means the X-axis, "1" means the Y-axis and "2" means the Z-axis.

@Value: (The data type can be "DINT", "UDINT", "DWORD" and "LINT")
It returns the current counter or encoder value. (Refer the Appendix A for the range of values)

Q: (Data type: "BOOL")
Counting state. "TRUE": Counting ; "FALSE": Stopped.

4.9.4 COUNTER_STATE

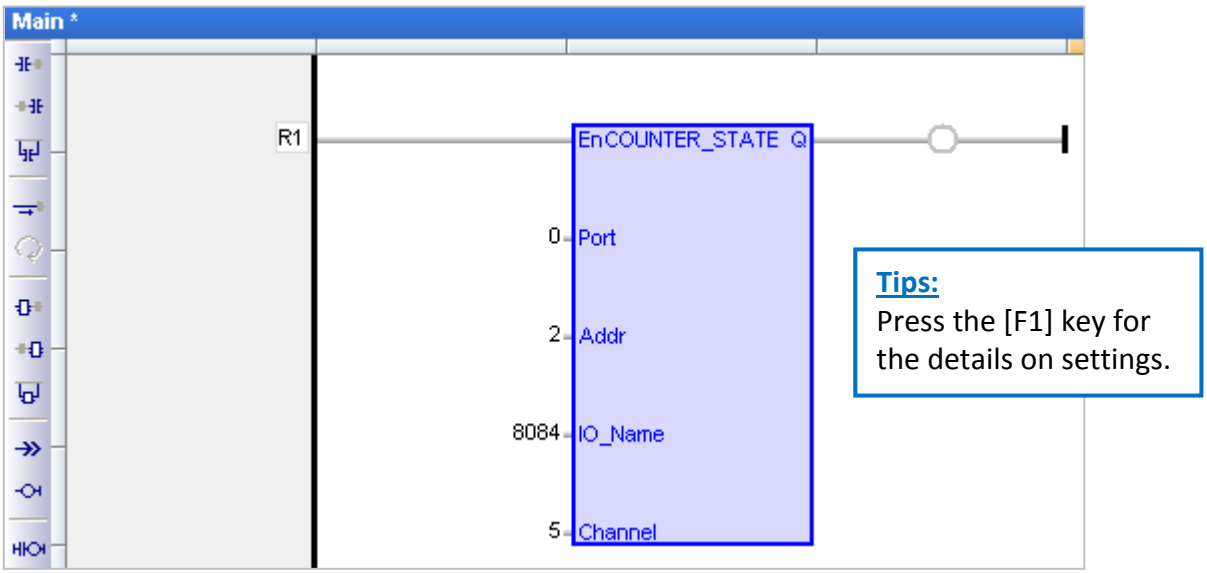
For example: Using the I-8084W module in the PAC's slot2 and to get the counting status of the channel 5.

ST program:

```
TMP_BOOL := Counter_State (0, 2, 8084, 5) ;
```

Note: First, add a "TMP_BOOL" BOOL variable in the Variable Area.

LD program:



Port: (Data type: "DINT")
For a module in the slot 0 to 7 of the PAC, set it as "0". For a DCON module connected to a serial COM port, can be "1 to 37" (depends on PAC, means COM1 to COM37).

Addr: (Data type: "DINT")
For a module in the slot 0 to 7 of the PAC, set it as the slot number (0 to 7).
For a module connected to a serial COM port, set it as the Net-ID address of the module (1 to 255).

IO_Name: (Data type: "DINT")
The name of relative Counter/Encoder module, it can be set to "8084", "8093", "87084", "87082", "7083" and "7080".

Channel: (Data type: "DINT")
The channel No. of the Counter/Encoder module, it can be set to "0", "1", and so on, depends on the module. For example, when using the I-8093W module, "0" means the X-axis, "1" means the Y-axis and "2" means the Z-axis.

Q: (Data type: "BOOL")
"TRUE": OK ; "FALSE": Error.

4.9.5 COUNTER_RESET

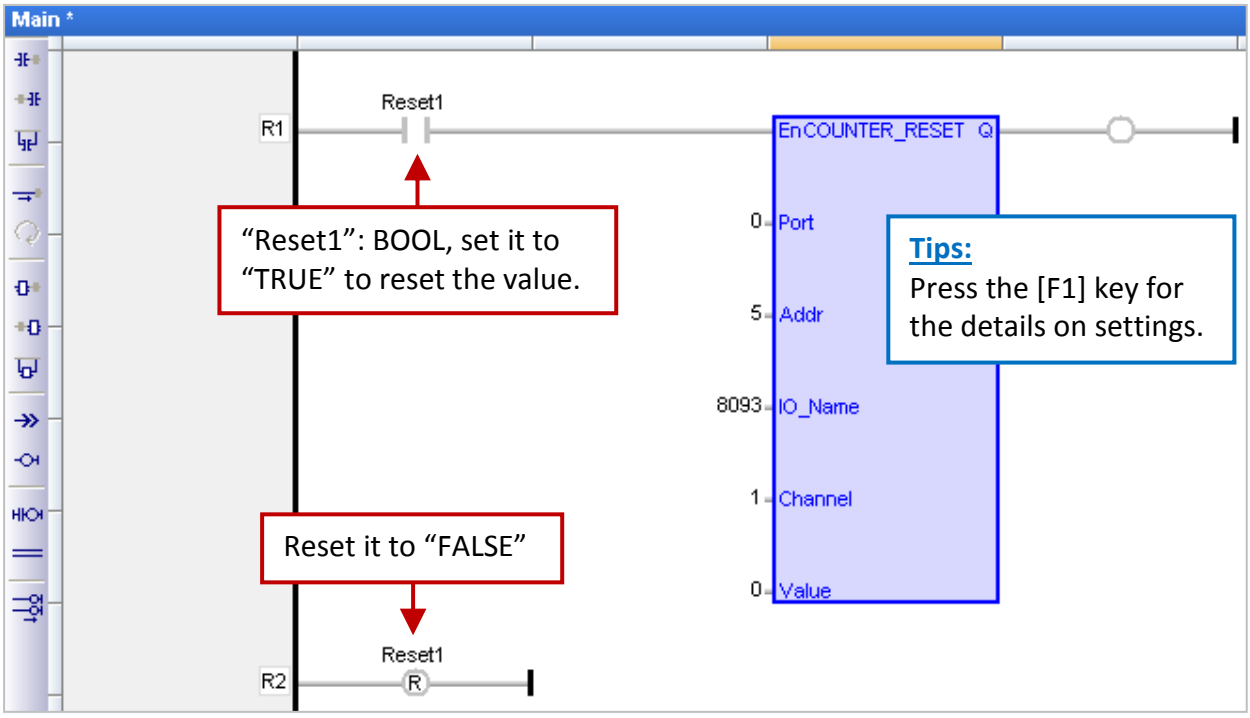
For example: Using the I-8093W module in the PAC's slot5 and reset the Encoder value of the Y-axis as "0".

ST program:

```
IF   Reset1 = TRUE   THEN
  Reset1  :=  FALSE ;
  TMP_BOOL  :=  Counter_Reset (0, 5, 8093, 1, 0) ;
END_IF ;
```

Note:
First, add two BOOL variables ("Reset1", "TMP_BOOL") in the Variable Area.

LD program:



Port: (Data type: "DINT")
For a module in the slot 0 to 7 of the PAC, set it as "0". For a DCON module connected to a serial COM port, can be "1 to 37" (depends on PAC, means COM1 to COM37).

Addr: (Data type: "DINT")
For a module in the slot 0 to 7 of the PAC, set it as the slot number (0 to 7).
For a module connected to a serial COM port, set it as the Net-ID address of the module (1 to 255).

IO_Name: (Data type: "DINT")
The name of relative Counter/Encoder module, it can be set to "8084", "8093", "87084", "87082", "7083" and "7080".

Channel: (Data type: "DINT")
The channel No. of the Counter/Encoder module, it can be set to "0", "1", and so on, depends on the module. For example, when using the I-8093W module, "0" means the X-axis, "1" means the Y-axis and "2" means the Z-axis.

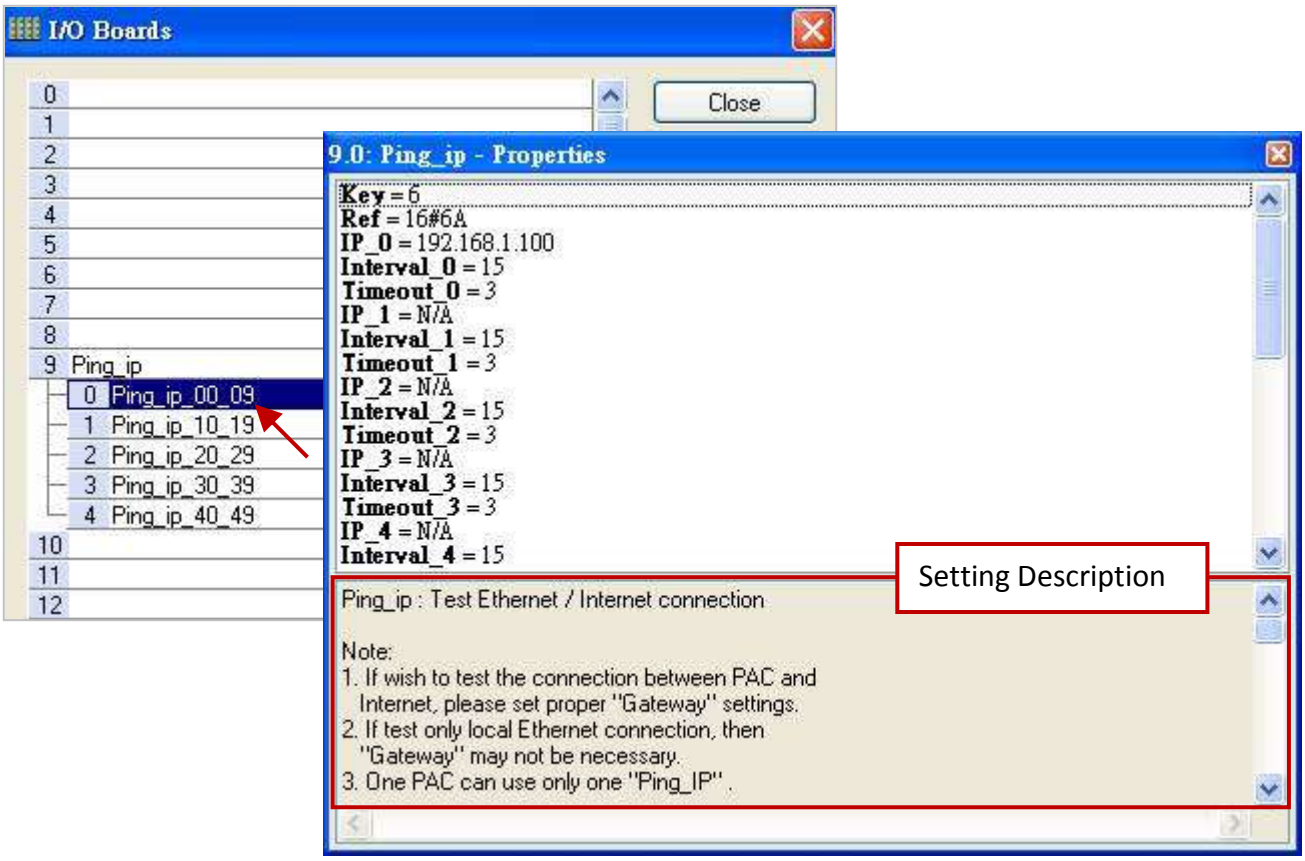
Value: (The data type can be "DINT", "UDINT", "DWORD", and "LINT")
The new Counter or Encoder value wish to set.

Q: (Data type: "BOOL") "TRUE": OK ; "FALSE": Error.

4.10 Ping_ip (Test an Ethernet/Internet Connection)

The "Ping_ip" function is used to test if the connection of the remote Ethernet/ Internet device is working properly. (It supports a max. of 50 IP settings.) If not familiar with the way to add this I/O board, see the Chapter4 (P4-1).

1. Mouse double-click the "Ping_ip" to open the "Properties" window, and then to see the setting description.
   Note: Using the slot 8 or above No. because the slot 0 to 7 are reserved for the real I/O module.



Note:

1. If wish to test the connection between PAC and Internet, please set proper "Gateway" settings.
2. If test only local Ethernet connection, then "Gateway" may not be necessary.
3. One PAC can use only one "Ping_IP". (Don't use two or more)
4. When Ping success, return Boolean channel as TRUE.
5. When Ping fails, it will try one more time. If still fail, then return Boolean channel as FALSE.

Parameters:

IP_01 to IP_49: (Data type: "STRING")

The IP address of targets. Set as 'N/A' if wish to disable it.

For example, 192.168.1.100 or 52.19.125.242 or N/A.

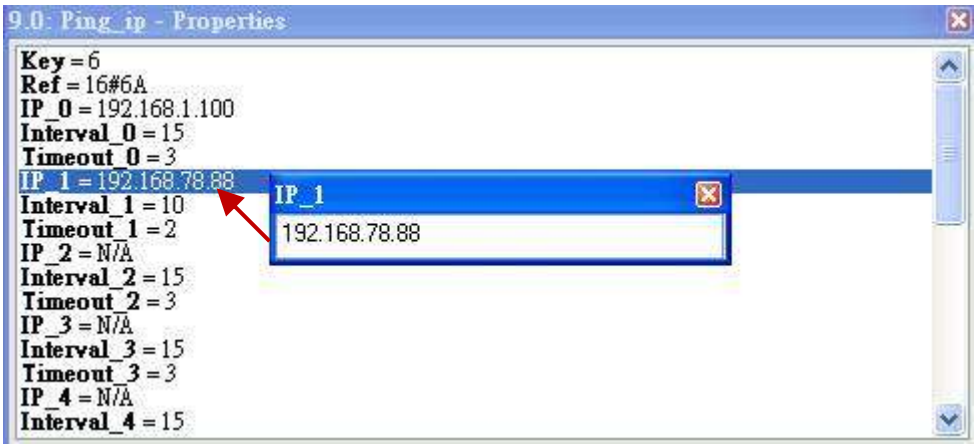
Interval_01 to Interval_49: (Data type: "DINT")

The unit is second. The interval to send one "ping" command. Value can be 6 to 86,400 seconds. Setting smaller than 6 will use as 6. Setting greater than 86400 (24 hours) will use as 86400.

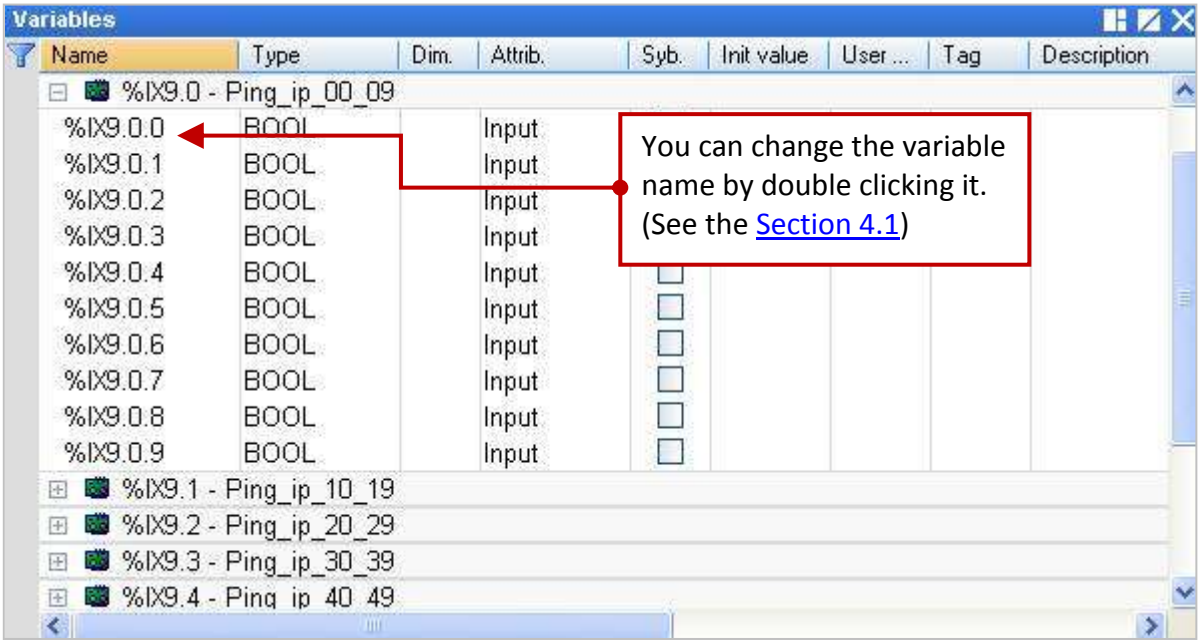
Timeout_01 to Timeout_49: (Data type: "DINT")
The unit is second. The timeout settings of the "ping" command. Value can be 2 to 30 seconds. Setting smaller than 2 will use as 2. Setting greater than 30 will use as 30.

Note: The "Interval_xx" value should be **at least triple** of the "Timeout_x" value. Or the PAC will use the "Interval_x" value as a triple of the "Timeout_x" value.
For example, if "Timeout_00" is set as 10 however "Interval_00" is set as 20, then PAC will use "Interval_00" as 30.

2. Double click the item and fill in a value, then press the "Enter" key to complete the setting.



3. After linking the "Ping_ip" in the "I/O Boards" window, it will auto add 50 "BOOL" input variables in the "Variables" window. When the Win-GRAF connects the PAC, it will display the online status.
   True: The connection is ok.
   FALSE: Connection failed or cable problem.

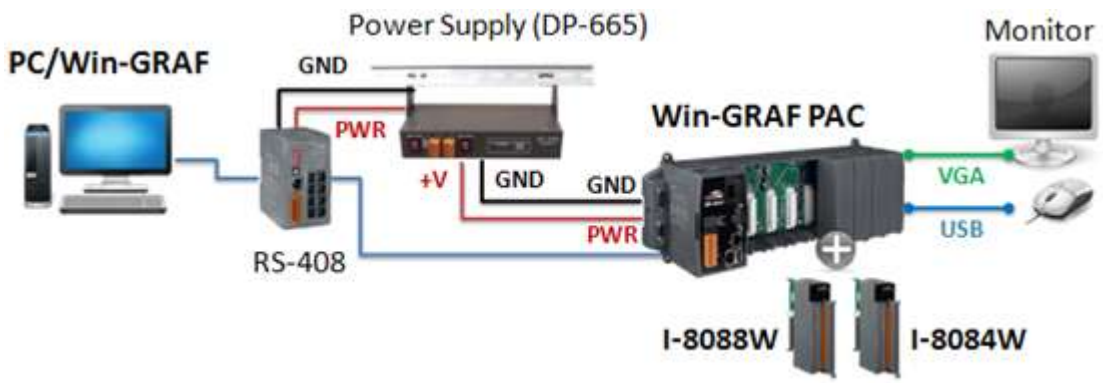


4.11 I-8088W (8-channel PWM Output Module)

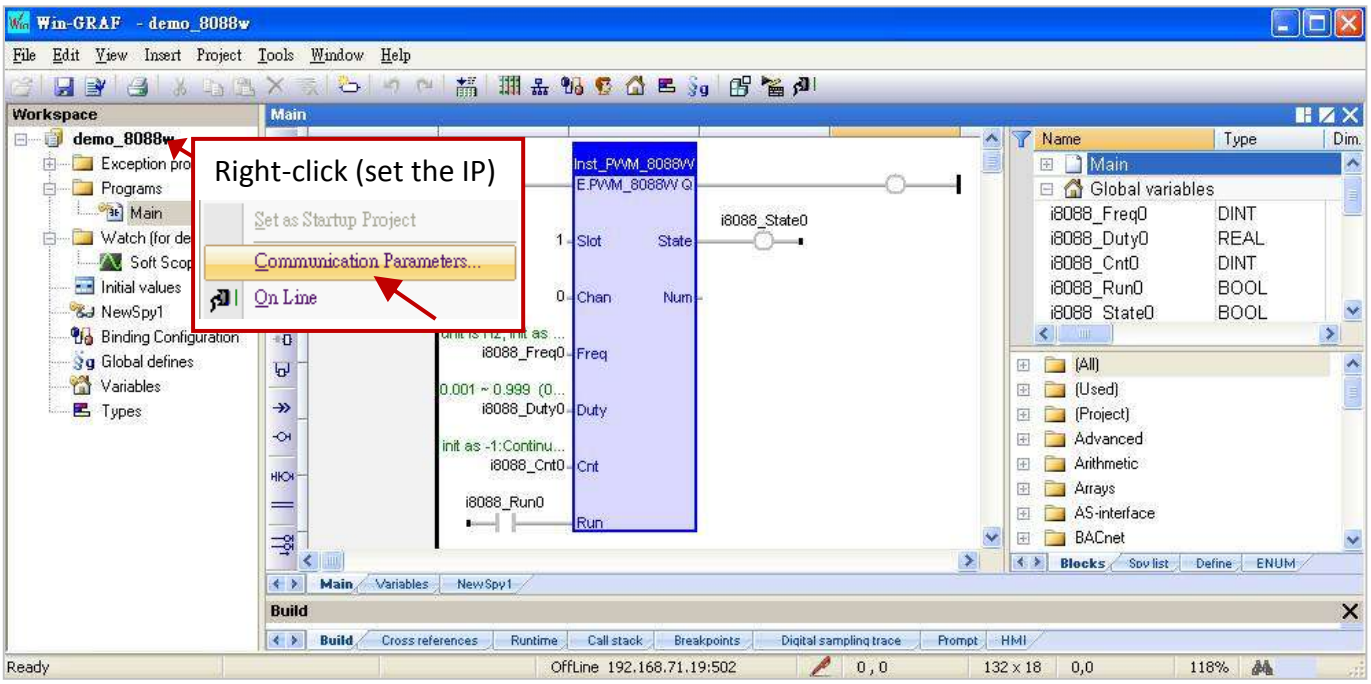
The I-8088W is an 8-channel PWM (Pulse Width Modulation) output module. The duty range (Duty = High / (High + Low)) of the PWM output signal can be from 0.1% to 99.9%. Its output frequency in the Win-GRAF PAC is from 1 Hz to 500 KHz. The I-8088W support two PWM output modes, one is the "Continuous" mode. It outputs always. The other one is the "Burst" mode. It outputs the required pulse count and then stop. Please visit http://www.icpdas.com/products/Remote_IO/i-8ke/i-8088w.htm for other specifications.

Hardware Connection Diagram:

This example uses the I-8084W (Slot 2) to measure the frequency of the I-8088W (Slot 1) PWM output signal (the I-8084W is not necessary in the actual application). Then, connect the I-8088W's PWM output channel 0 (PW0) to the I-8084W's frequency input channel 0 (C0A+).

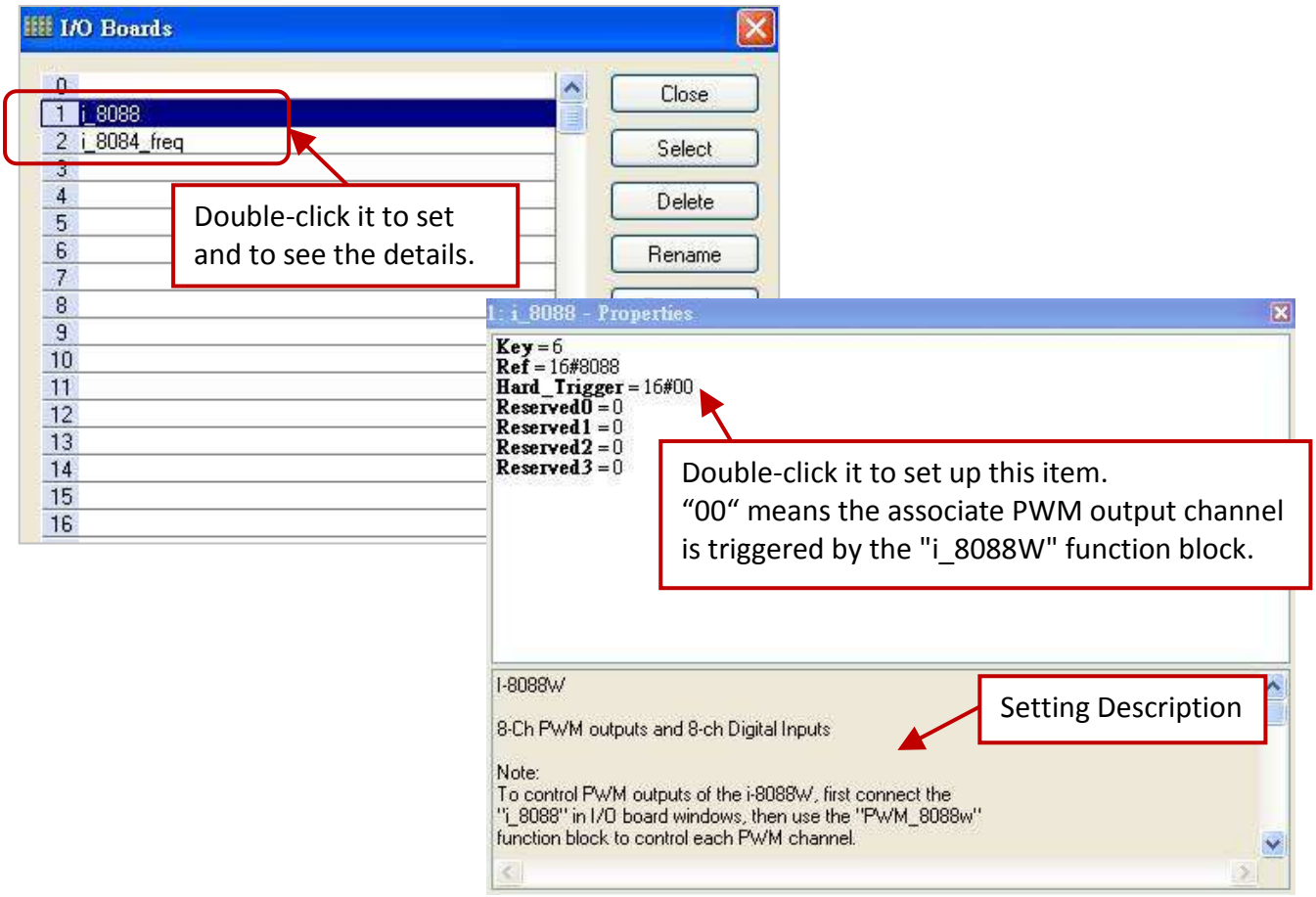


The demo project (demo_8088w.zip) that we will describe below is located in the shipment CD (\Napdos\Win-GRAF\demo-project) , refer the Chapter 12 to restore/open this project and set up the current PAC's IP address.

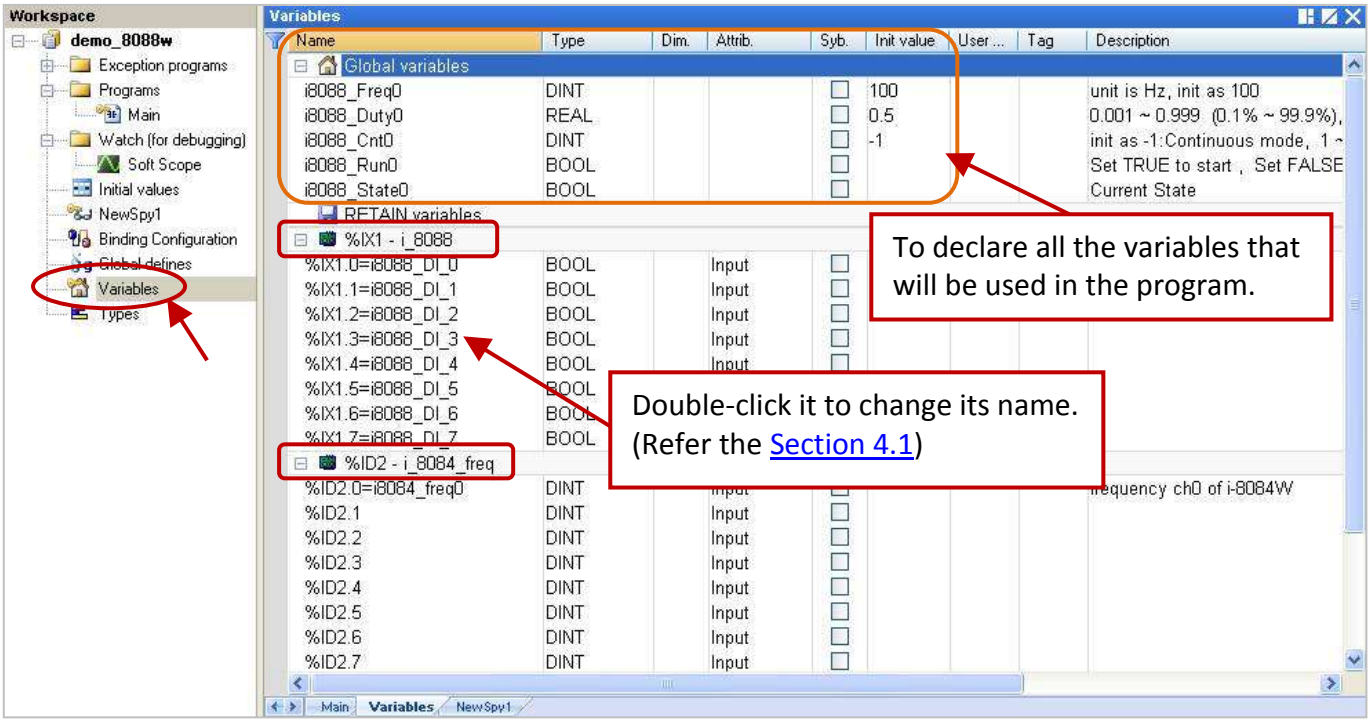


I/O Boards:

In this case, add the "i_8088" and the "i_8084_freq" to the corresponding I/O slot No. in the "I/O Boards" window (see the Chapter 4). Then, mouse double-click the Slot No. to open the "Properties" window and you can see the setting description to set this I/O board.

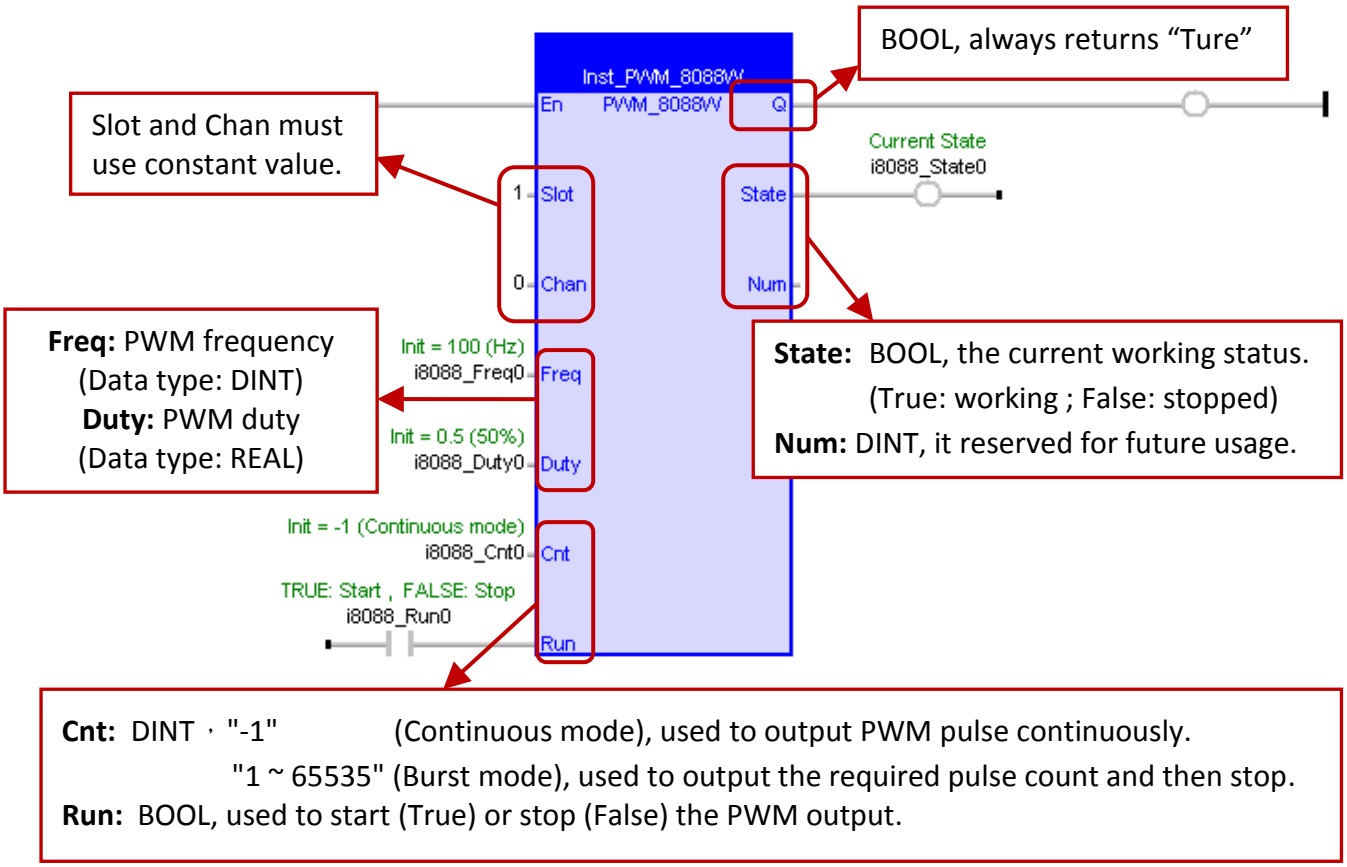


After linking the "i_8088" and the "i_8084_freq" I/O boards, it will auto add related variables in the "Variables" window (or Variables Area). And, you can also declare all the variables that will be used in the program here (refer the Section 2.3.1).



"PWM_8088W" Function Block:

Then, using the function block “PWM_8088W” to control the PWM output for each channel, such as the LD program below.



<u>Parameters:</u>

Slot: The used I/O slot No., and it must be a constant value, not a changeable value. In this case, the value is "1". (Data type: DINT)

Chan: The used I/O channel No., and it must be a constant value, not a changeable value. In this case, the value is "0". (Data type: DINT)

Freq: The PWM output frequency. (Data type: DINT ; Unit: Hz)
The value can be from 1 to 500,000 (i.e., 1 Hz to 500 KHz) .
In this case, the initial value is "100" Hz.

Duty: The PWM output. (Data type: REAL)
The value can be from 0.001 to 0.999 (i.e., 0.1 % to 99.9 %).
In this case, the initial value is 0.5 (i.e., 50 %).

Cnt: The output mode (Data type: DINT)
Continuous mode: set it as “-1” (in this case) to output PWM pulse continuously.
Burst mode: it can be from "1" to "65535", to output the required pulse count and then stop.

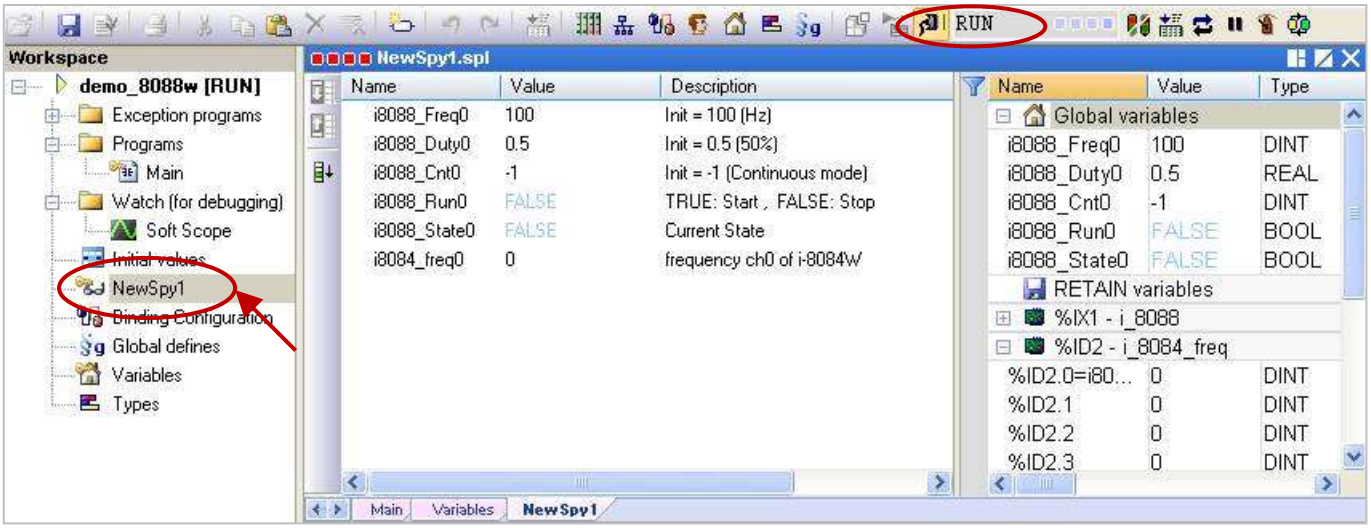
Run: Using a BOOL variable to trigger the PWM output. (True: Start ; False: Stop)

State: The current working status. (Data type: BOOL). (True: working ; False: stopped)

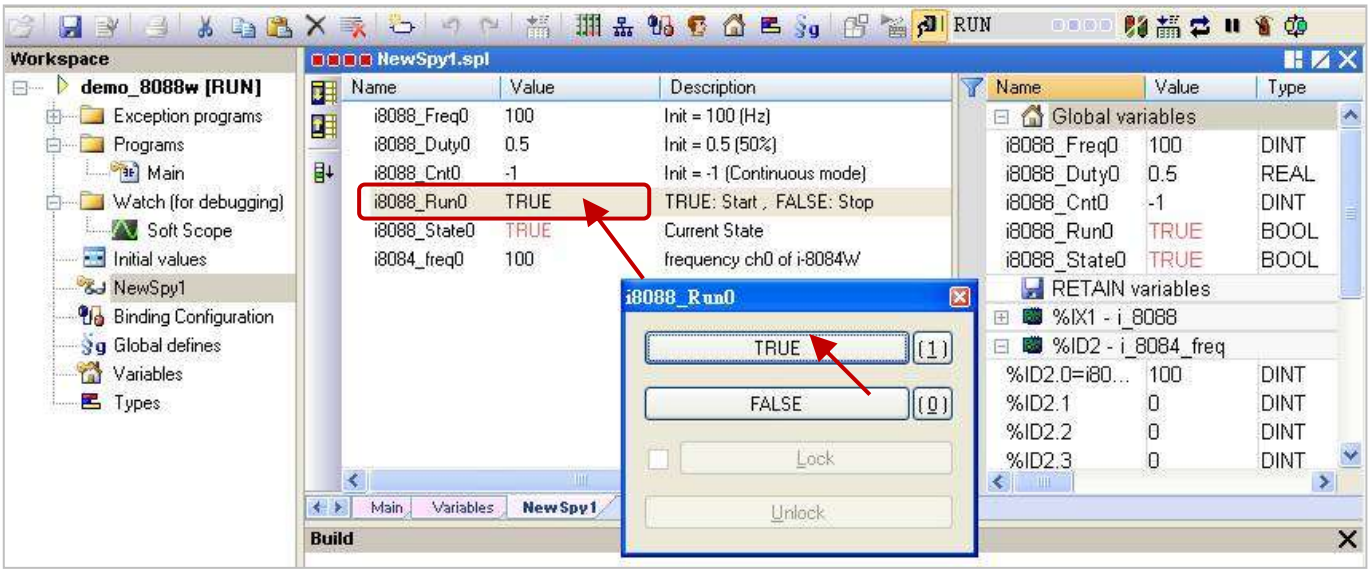
Test the program:

Before testing, make sure you have set the PAC's IP address and then compile and download this program to the PAC. (If not familiar with the operation, refer the Section 2.3.4 and Section 2.3.5.)

When connecting with the PAC, the SPY List (refer the Section 11.3) will show that the I-8088W's PWM initial frequency ("i8088_Freq0") is 100 Hz, the initial duty cycle ("i8088_Duty0") is 50%, using the Continuous mode ("i8088_Cnt0" = -1) and the currently measured frequency of the I-8084W is 0 Hz.



Now, set the "i8088_Run0" as "TRUE" to start the PWM output. At this time, the "i8088_State" will change from "FALSE" to "TRUE" and output a PWM signal to the I-8084W module, and then the value of the "i8084_Freq0" will change from 0 Hz to 100 Hz.



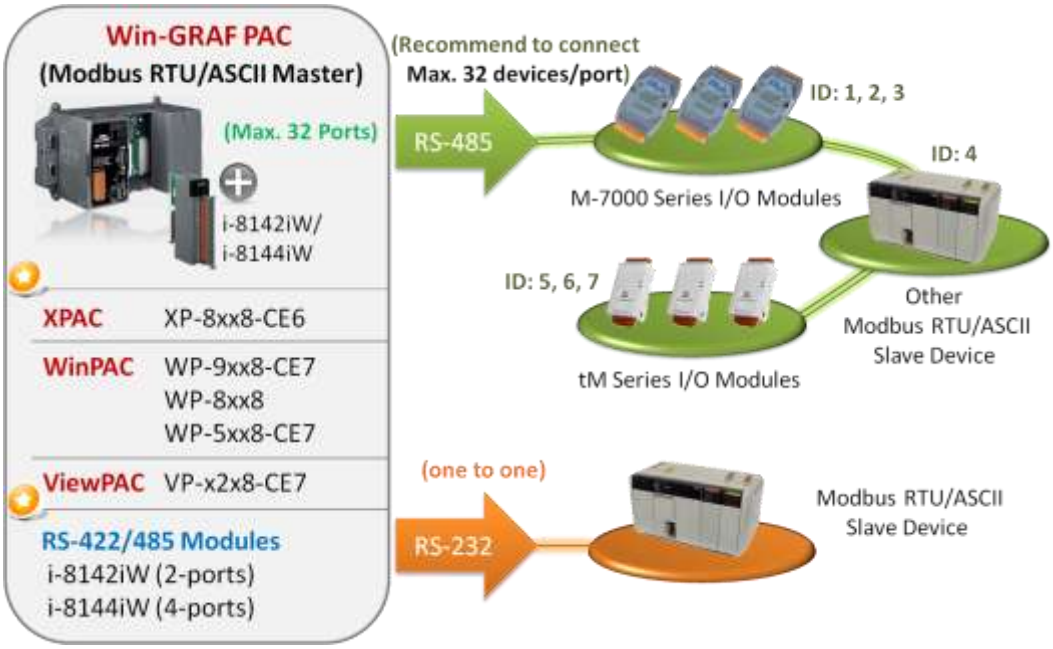
If set the "i8088_Cnt0" as 500 (Burst mode), the "i8084_Freq0" value will become 0 after the I-8088W outputs 500 PWM pulses. You can try to change the "i8088_Cnt0" value and then set the "i8088_Run0" as "TRUE" to view the changes of output.

Chapter 5 Modbus Master: connecting to Modbus Slave Devices

This chapter lists the way to enable the Win-GRAF PAC as a Modbus Master to connect Modbus RTU/ASCII Slave or Modbus TCP/UDP Slave devices. If you want to use one XV board in the WP-5xx8-CE7, refer the Section 5.1.6 to Section 5.1.11.

5.1 Enabling the Win-GRAF PAC as a Modbus RTU/ASCII Master (I/O & XV-board)

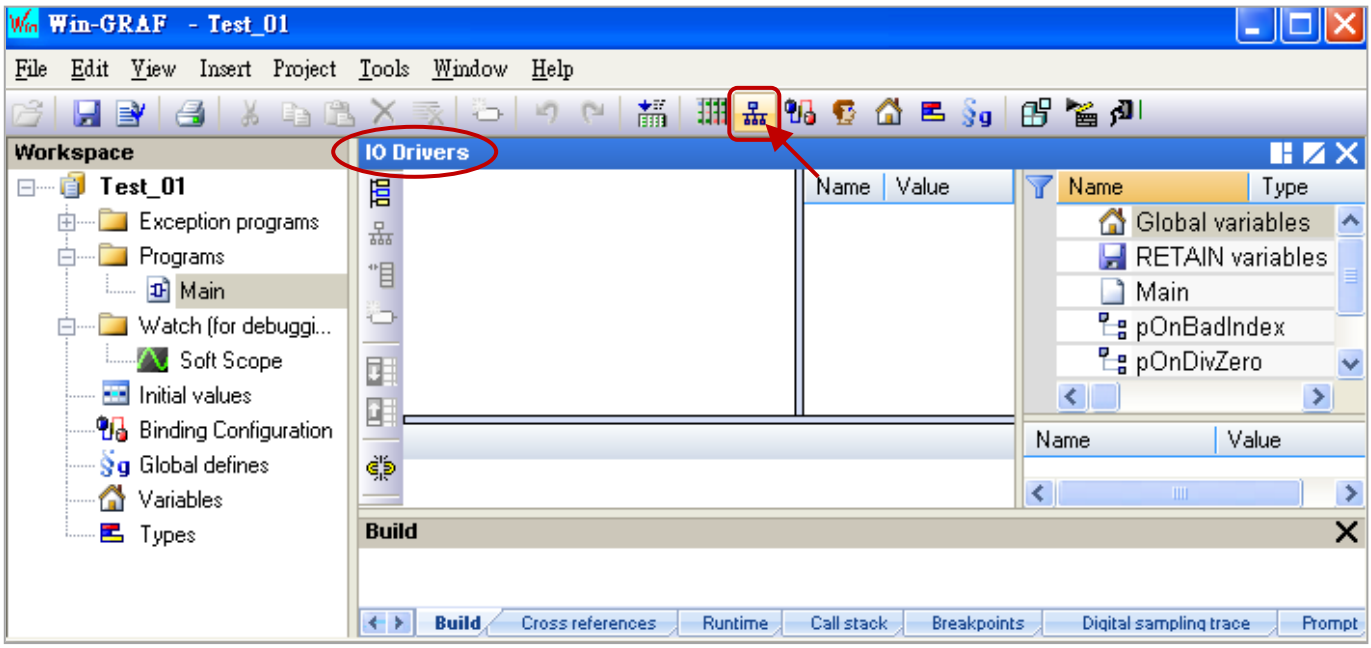
Application Diagram:



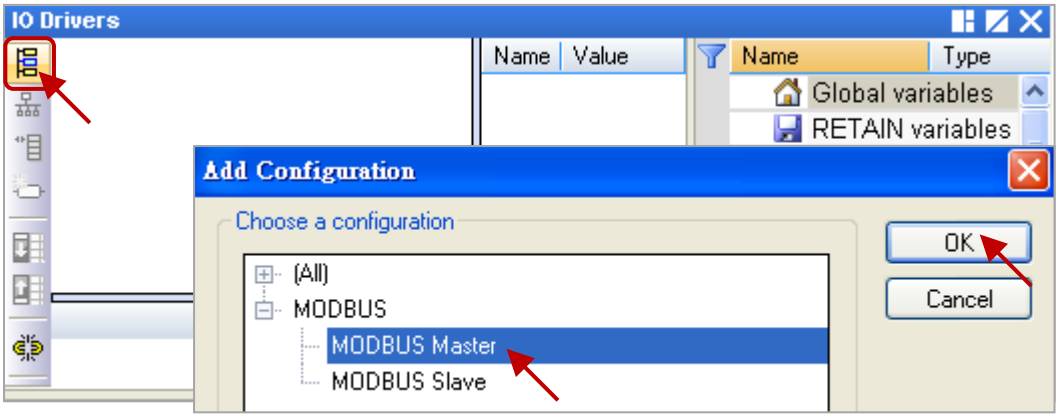
(Refer the P1-1 to view all PAC models)

Follow these steps:

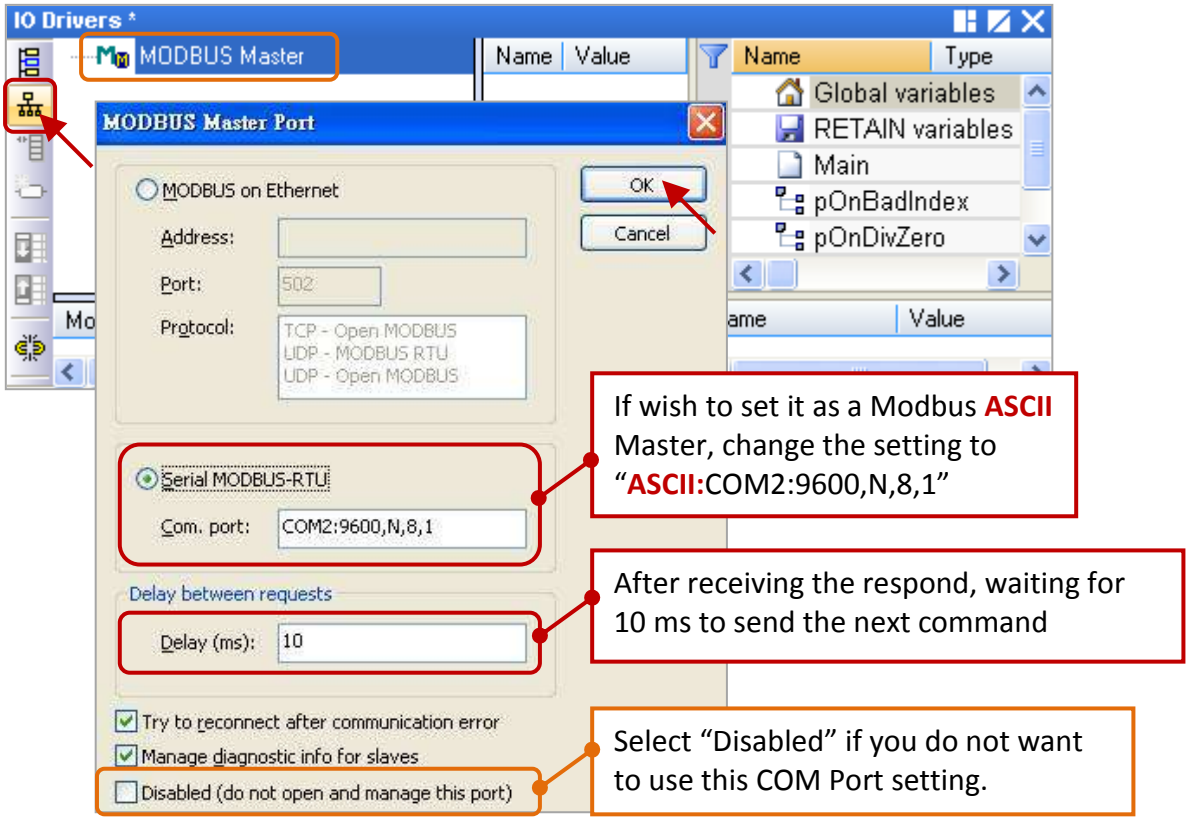
1. Mouse click the "Open Fieldbus Configuration" tool button to open "IO Drivers" window.



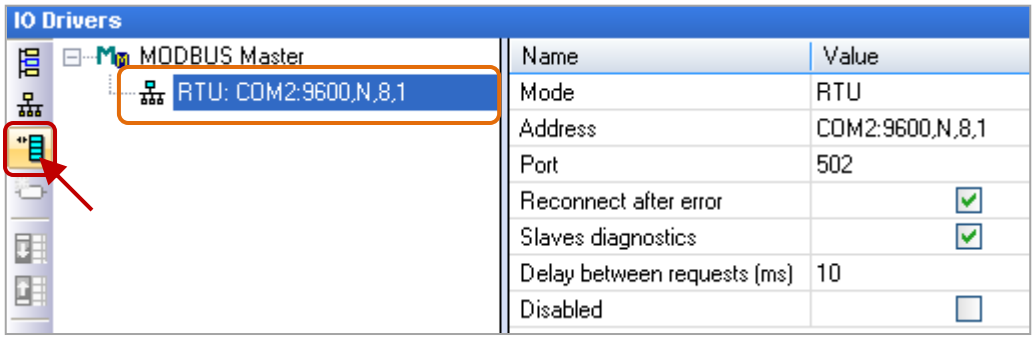
2. Click the “Insert Configuration” button on the left of the “IO Drivers” window, then click the “MODBUS Master” and “OK” to enable the Modbus Master setting.



3. Click the “Insert Master/Port” button on the left side to open the setting window. Then, select the “Serial MODBUS-RTU”, set COM Port (e.g., “COM2:9600,N,8,1”) and Delay time (recommended value: 10 ms, it can be 0 to 10000), and then click “OK”.



4. Click the “Insert Slave/Data Block” button on the left side to create a data block.



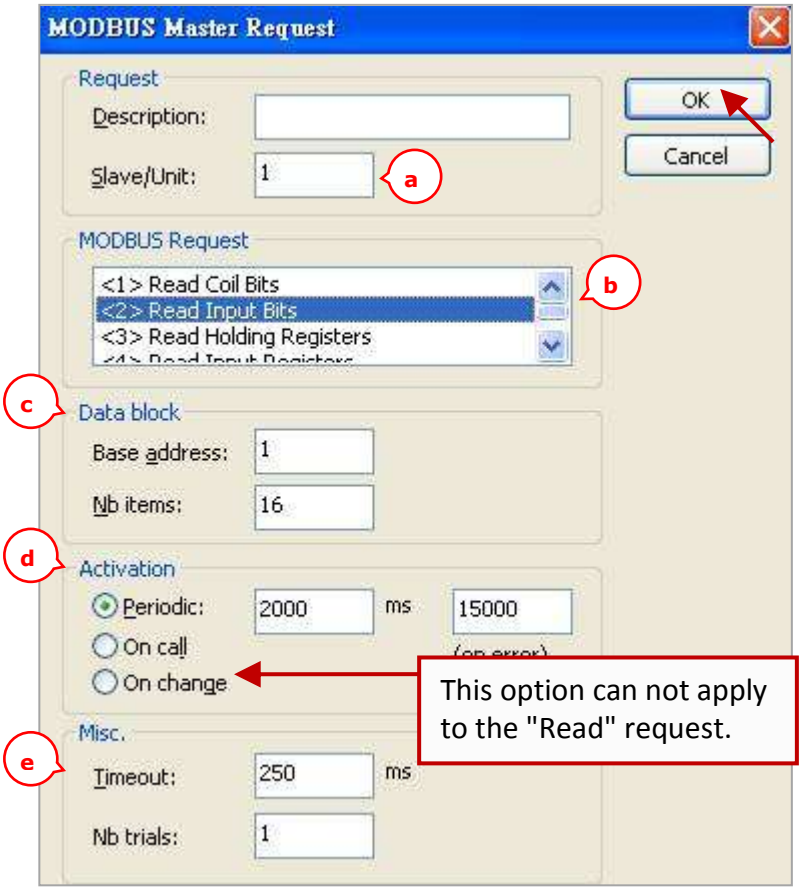
This table lists five data blocks, and each data block stands for one Modbus Master Request.

Item	Function Code	Modbus Request	Description
1	2	Read Input-bits	Read DI data
2	5	Write single coil-bit	Write DO data
3	4	Read Input Registers	Read AI data
4	6	Write single holding register	Write one AO data (16-bit)
5	16	Write Holding Registers	Write multiple AO data (16/32 bits)

Note: If you want to disable the Modbus RTU/ASCII Master port while the program is running, refer the Section 5.1.13 to use the "MBRTU_M_disable" function.

5.1.1 Read DI data

1. Completing all the following settings in the "MODBUS Master Request" window, and then click "OK".

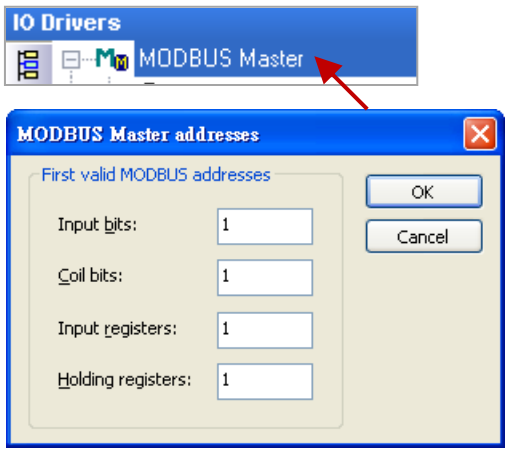


a. Slave/Unit:
Enter the Net-ID of the Slave device.
(In this case, the Net-ID is "1").

b. MODBUS Request:
Select "<2> Read Input Bits" option.

c. Base address:
Start from "1" by default.
Nb items:
The number of DI signals to read.
(In this case, the number is "16").

Note:
If you want to change the "Base address", right-click the "MODBUS Master" and then select the "MODBUS Master Addresses" to modify the value.



d. Activation: The way to send the Modbus request.

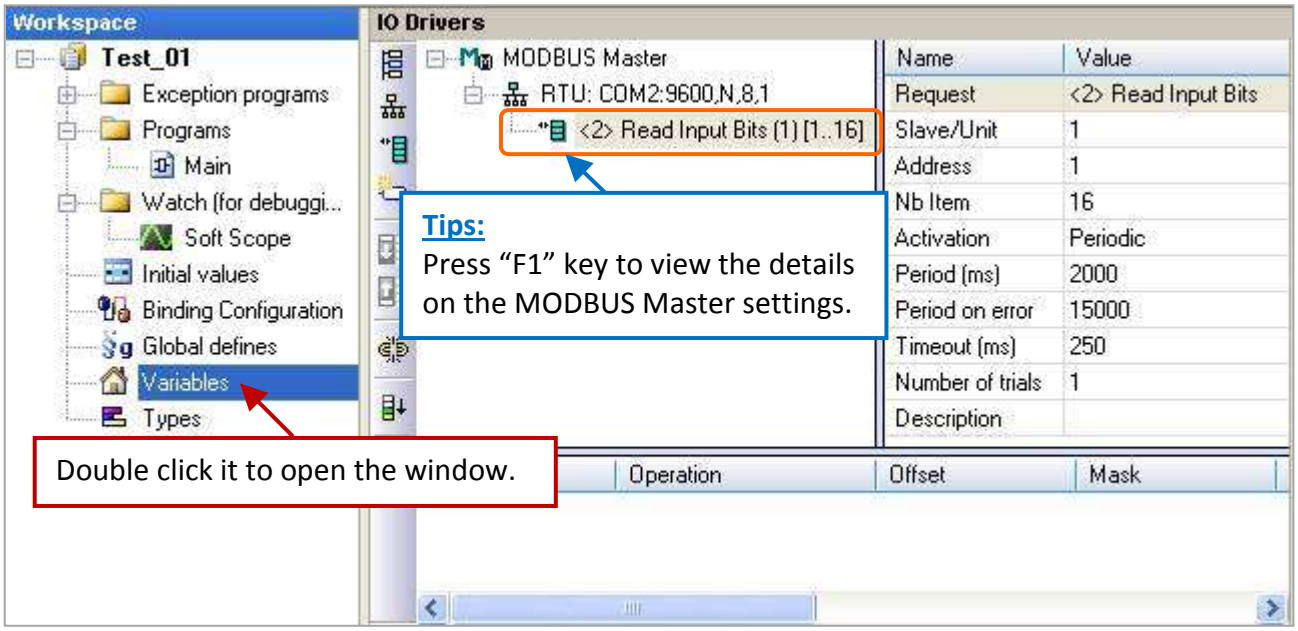
Periodic: Sending the request periodically.
(In this case, to send once every two seconds.)
"on error" means the next sending time when an exception occurred (e.g., 15 seconds).

On call: The request is activated when a program call to send it

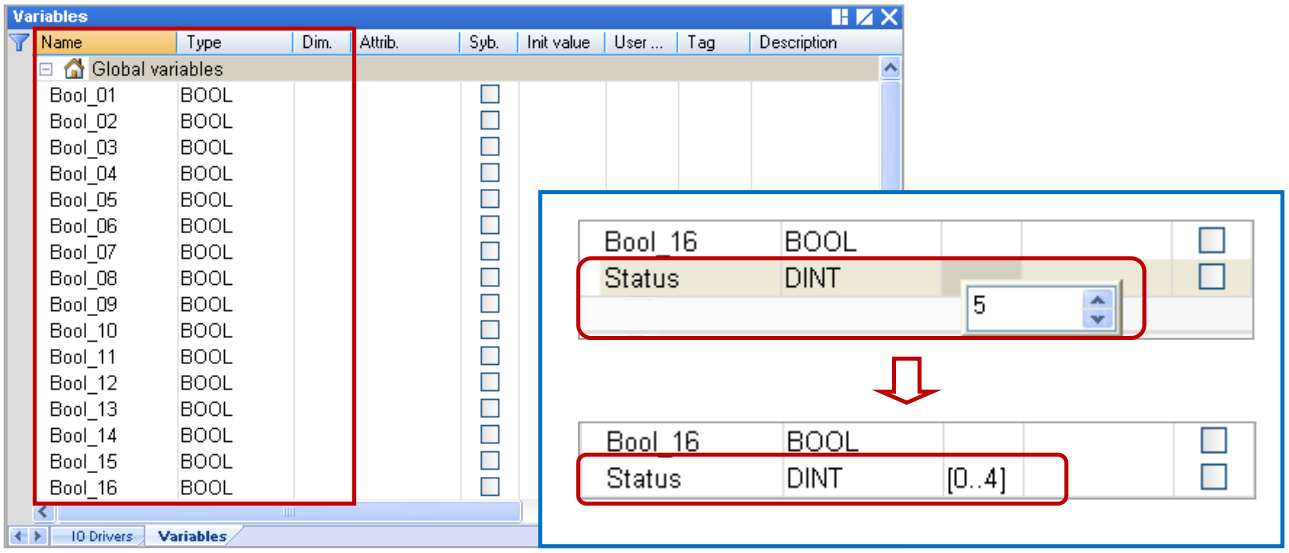
On change: In case of a write request, means that the request is activated each time any variable changed.

e. Timeout: Set a timeout value. (When time-out occurred, it will show the defined error code.)
(The recommended value for the Modbus RTU/ASCII device is 200 to 1000 ms. E.g., 250 ms)

2. Next, open the “Variables” window and then declare variables that are available for the program.



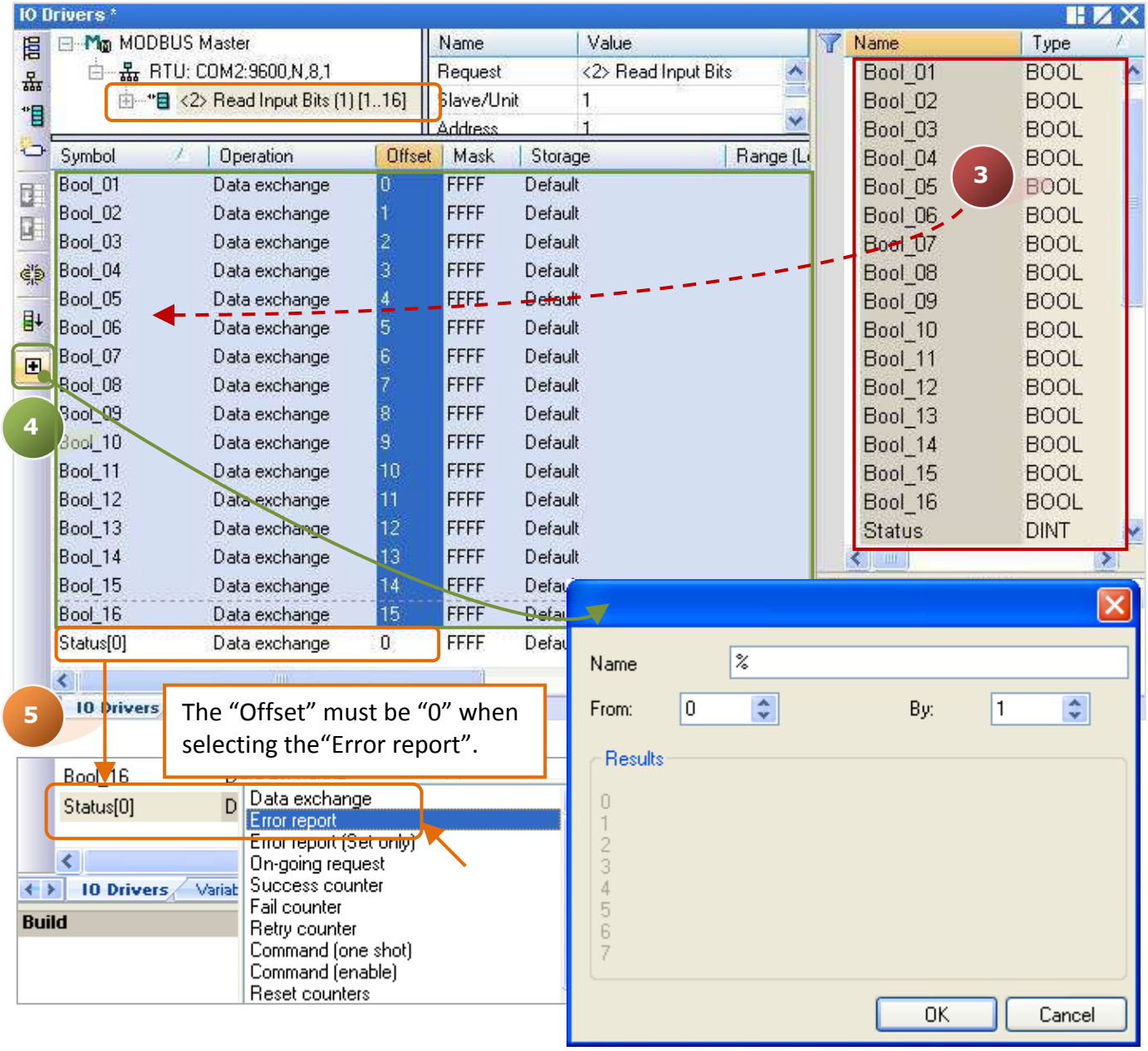
Declaring 16 variables to read data (Name: “Boo_01 to Boo_16”; Type: BOOL) and one array variable to record the state of data access (Name: “Status"; Dim.: 5; Type: DINT). Refer the Sectin 2.3.1 for the way to declare variables, and the figure below shows defined variables.



3. In the "IO Drivers" window like the figure below, drag all required variables in the Variables Area (i.e., “Bool_01” to “Bool_16” and “Status”) and drop them to the “Symbol” area in the first data block.
   Note: The “Status” is an array variable, so, the Status[0] to Status[4] will show on the “Symbol” area.
   Click the “Del” key to delete the Status[1] to Status[4].

4. Next, select “Offset” field from “Boo_01” to “Boo_16” and then click the “Iterate Property” button on the left side to set the “Offset” value (From: “0” ; By: “1”, refer the Section 3.1 – Step8).

5. In the “Operation” field, set the “Status[0]” as “Error report” which means the return value is an “Error Code” if a read error occurred and the value will be reset to “0” if read successfully.

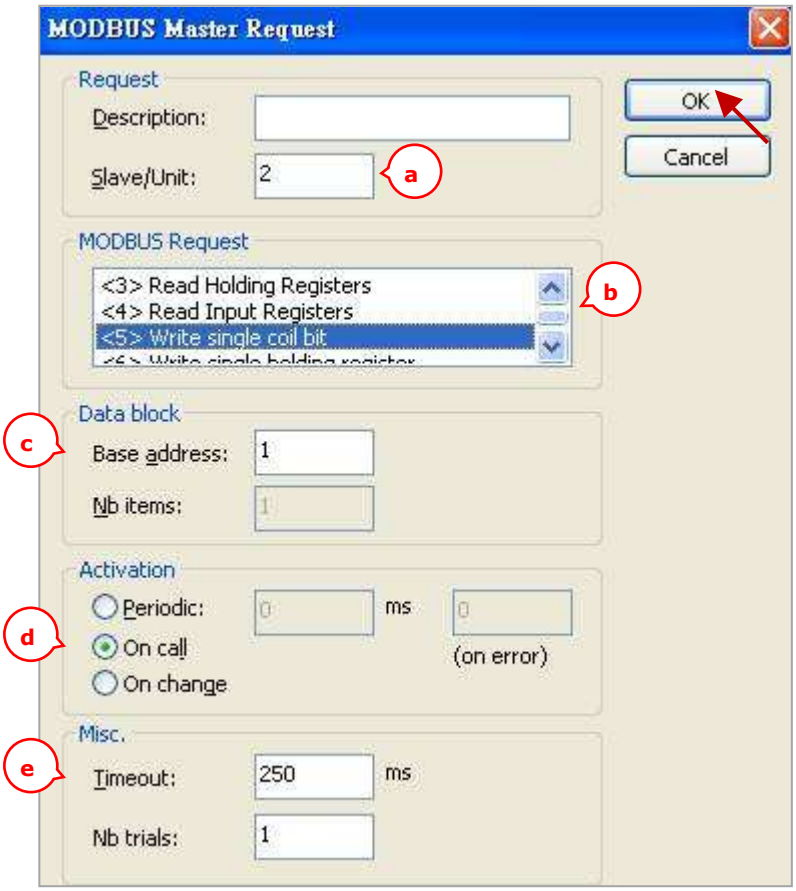


You can also press “F1” in this “IO Drivers” window to see details on Modbus Master Configuration.

Error Code	Description	Error Code	Description
0	The communication is OK.	8	Data Parity Error.
1	MODBUS function not supported.	10	Invalid gateway path.
2	Invalid MODBUS address.	11	Gateway target failed.
3	Invalid MODBUS value.	128	Communication timeout.
4	MODBUS Server failure.	129	Bad CRC16.
6	Server is busy.	130	RS-232 communication error.

5.1.2 Write DO Data

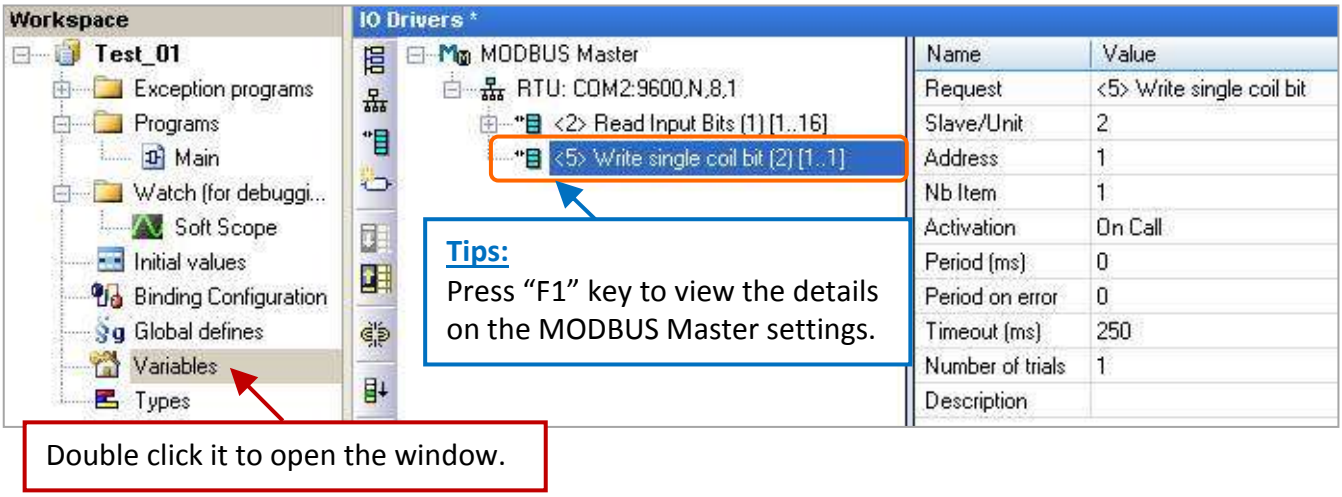
1. Using the same way in the Section 5.1 - Step 4 to create the second data block and completing all the following settings in the "MODBUS Master Request" window, and then click "OK".



In this example

a. Slave/Unit:
   Enter the Net-ID of the Slave device. (e.g., the Net-ID is "2").
b. MODBUS Request:
   Select "<5> Write single coil bit".
c. Base address:
   Start from "1" by default.
   (Refer the Section 5.1.1 to change it.)
d. On call:
   The request is activated when a program call to send it
   (Refer the Section 5.1.1 for details)
e. Timeout: Set a timeout value.
   When time-out occurred, it will show the defined error code. (The recommended value for the Modbus RTU/ASCII device is 200 to 1000 ms. In this case the value is 250 ms.)

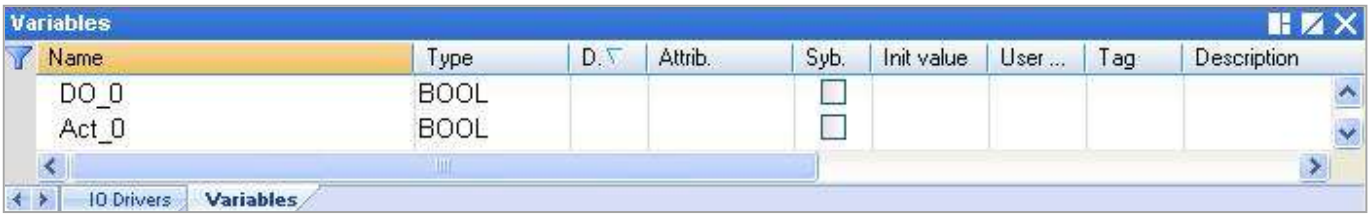
2. Next, open the "Variables" window and then declare variables that are available for the program.



Add two boolean variables in the "Variables" window (refer the Section 2.3.1 for declaring variables).

Variable name	Data type	Description
DO_0	BOOL	Used to Write digital output data.
Act_0	BOOL	In this case, choose the "On call" way to write data that means using a variable to call it.

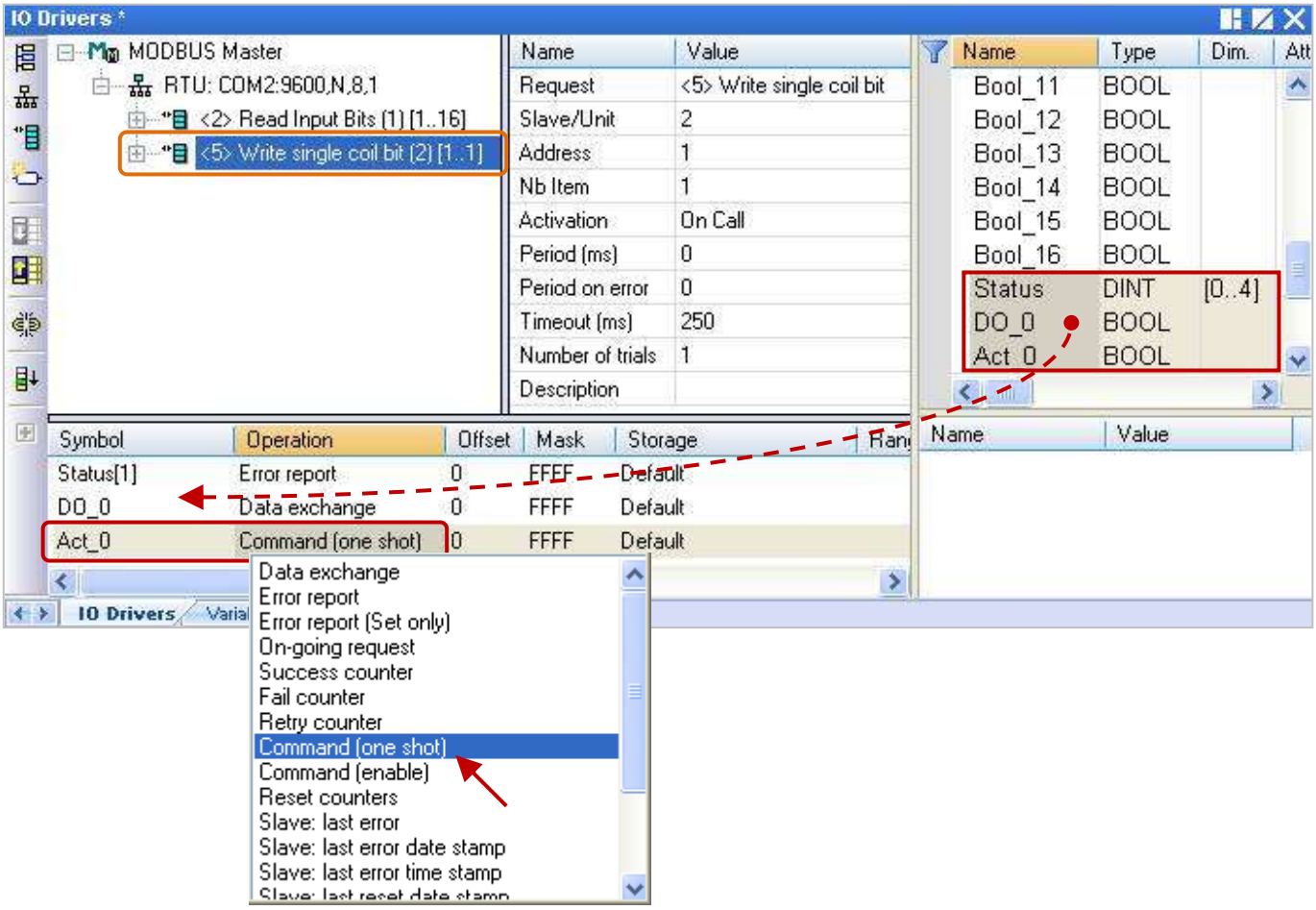
After completing the settings, the defined variables show as below:



3. In the "IO Drivers" window, drag variables - "DO_0", "Act_0" and "Status" (that created in the Section 5.1.1) from the Variables Area to the Symbol Area in the second data block.
   Note: The "Status" is an array variable. When you drag "Status" into the Symbol Area, it will show "Status[0]" to "Status[4]", simply *press "Del" key to delete "Status[0]" and "Status[2] to [4]".*

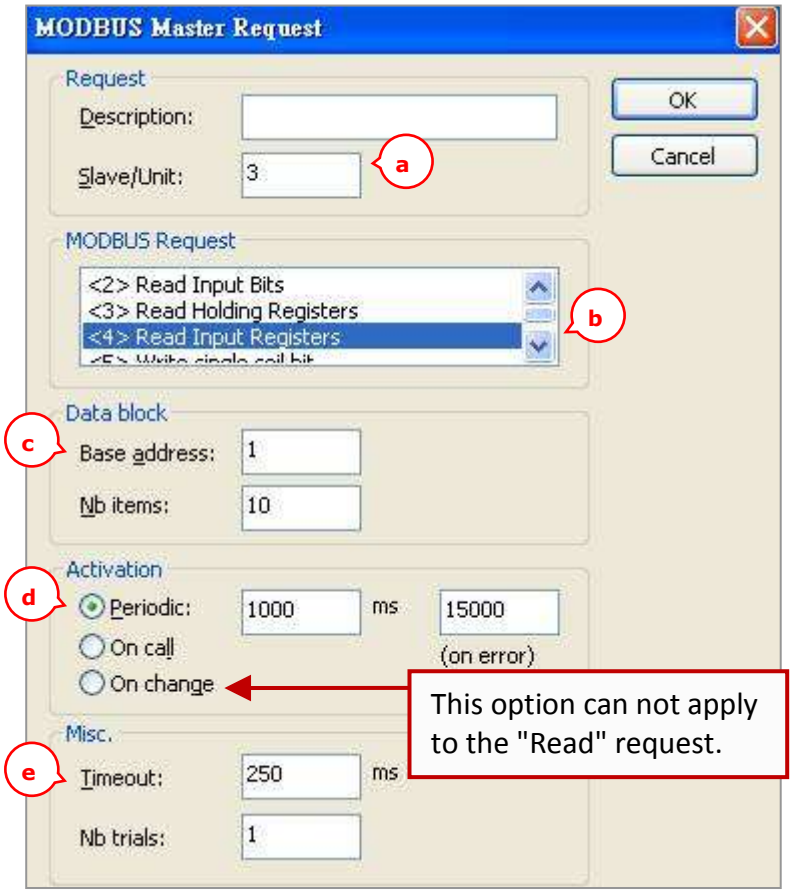
4. Set the "Operation" field of the "Status[1]" as "Error report" (that means this variable will be set to an error code when a read error occurs, or reset it to "0" when a read request is successful).
   Press the "F1" key to see the description of the Modbus Master Configuration and move to the title "Status and command variables" to know related commands and error codes.

5. Set the "Operation" field of "Act_0" as "Command (one shot)" (that means the request will be sent only once when "Act_0" is set to "TRUE". Then, this "Act_0" will auto reset to "FALSE").
   The "Command (Enable)" means the request is sent continuously as long as the "Act_0" is "TRUE". So, users can set the "Act_0" to "FALSE" to stop sending command.



5.1.3 Read AI Data

1. Using the same way in the Section 5.1 - Step 4 to create the third data block and completing all the following settings in the “MODBUS Master Request” window, and then click “OK".



In this example

a. Slave/Unit:
Enter the Net-ID of the Slave device.
(e.g., the Net-ID is “3”).

b. MODBUS Request:
Select “<4> Read Input Registers”.

c.

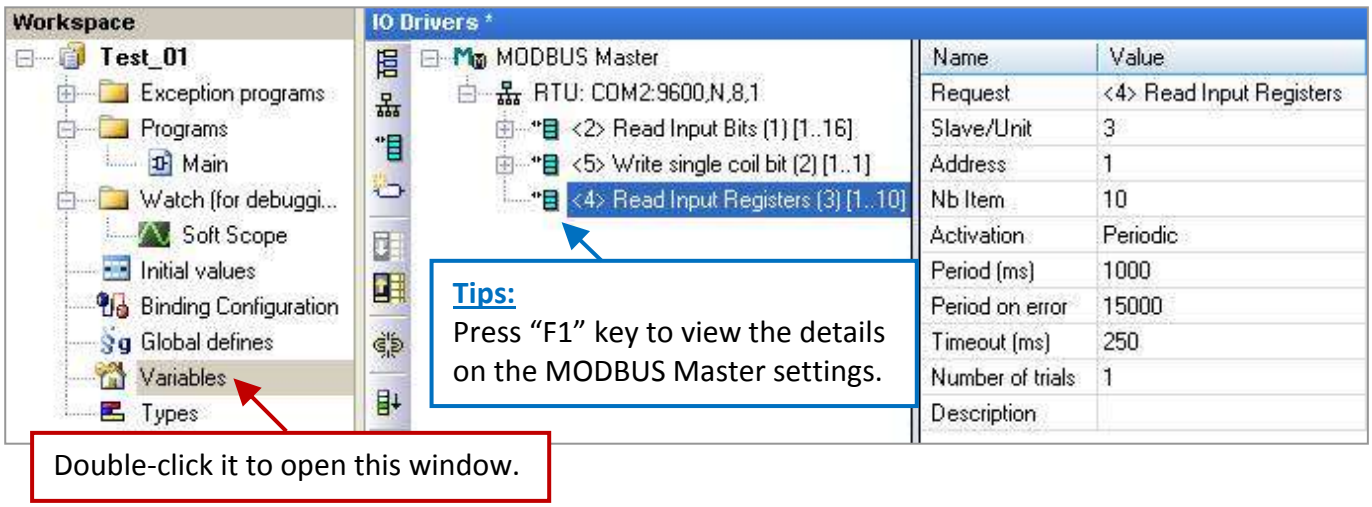
d. Base address:
Start from “1” by default.
(Refer the Section 5.1.1 to change it.)
Nb items:
The number of AI signals to write.
(In this case, the number is “10”).

e.

d. Periodic: (Refer the Section 5.1.1)
Sending the request periodically.
(In this case, to send once per second.)
“on error” means the next sending time when an exception occurred (e.g., 15 seconds).

e. Timeout: Set a timeout value.
When time-out occurred, it will show the defined error code. (The recommended value for the Modbus RTU/ASCII device is 200 to 1000 ms. In this case the value is 250 ms.)

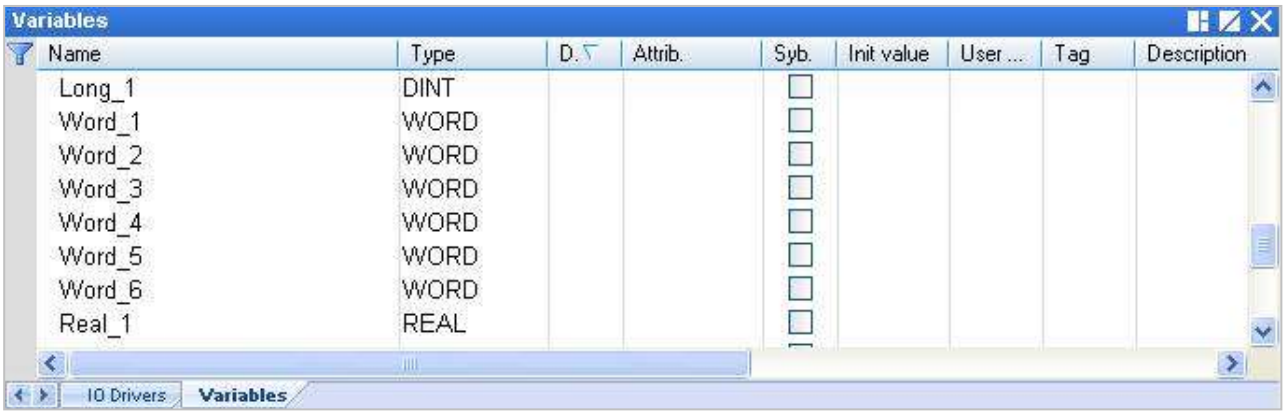
2. Next, open the “Variables” window and then declare variables that are available for the program.



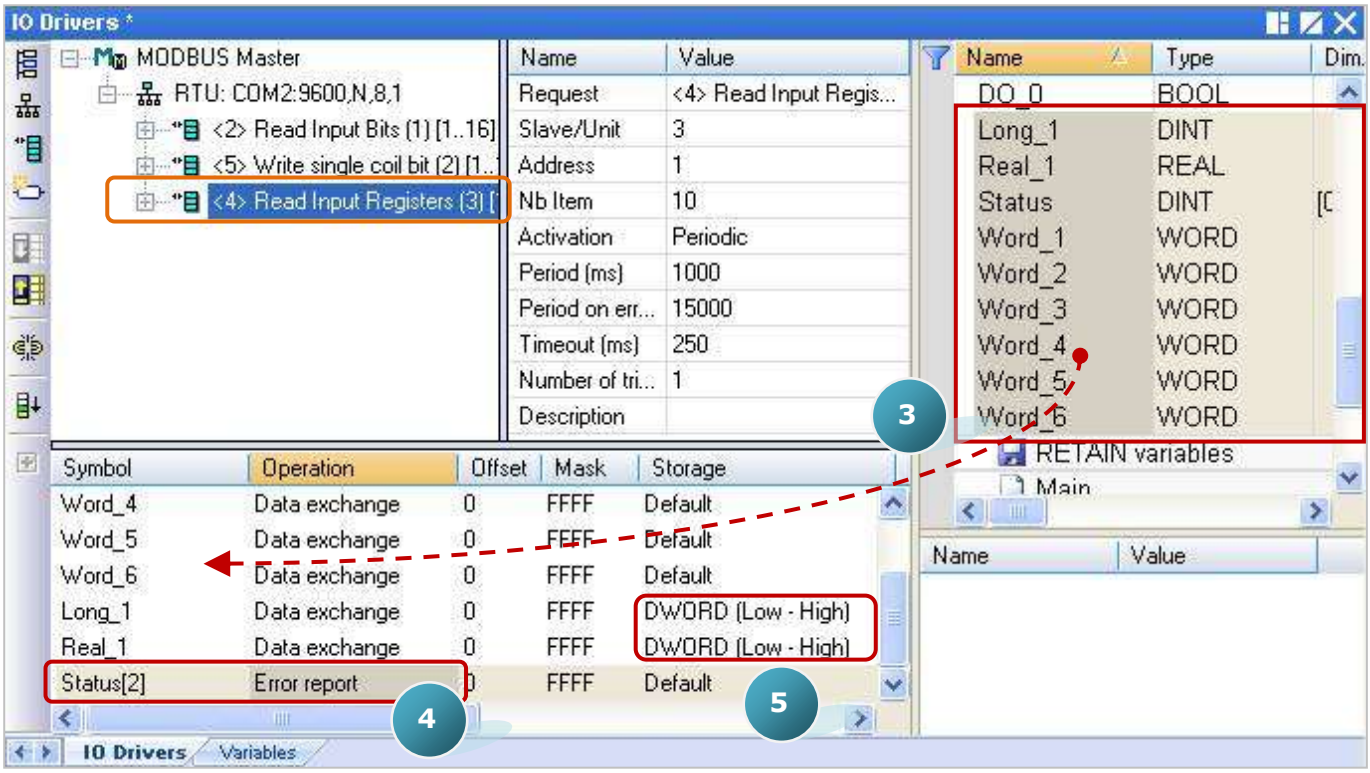
Follow the table below to add six Word (16-bit), one Double integer (32-bit) and one Real (32-bit) variables. (Refer the Section 2.3.1 for declaring variables).

Variable name	Data type	Description
Word_1 to Word_6	WORD	Used to Read AI data (16-bit).
Long_1	DINT	Used to Read AI data (32-bit).
Real_1	REAL	Used to Read AI data (32-bit).

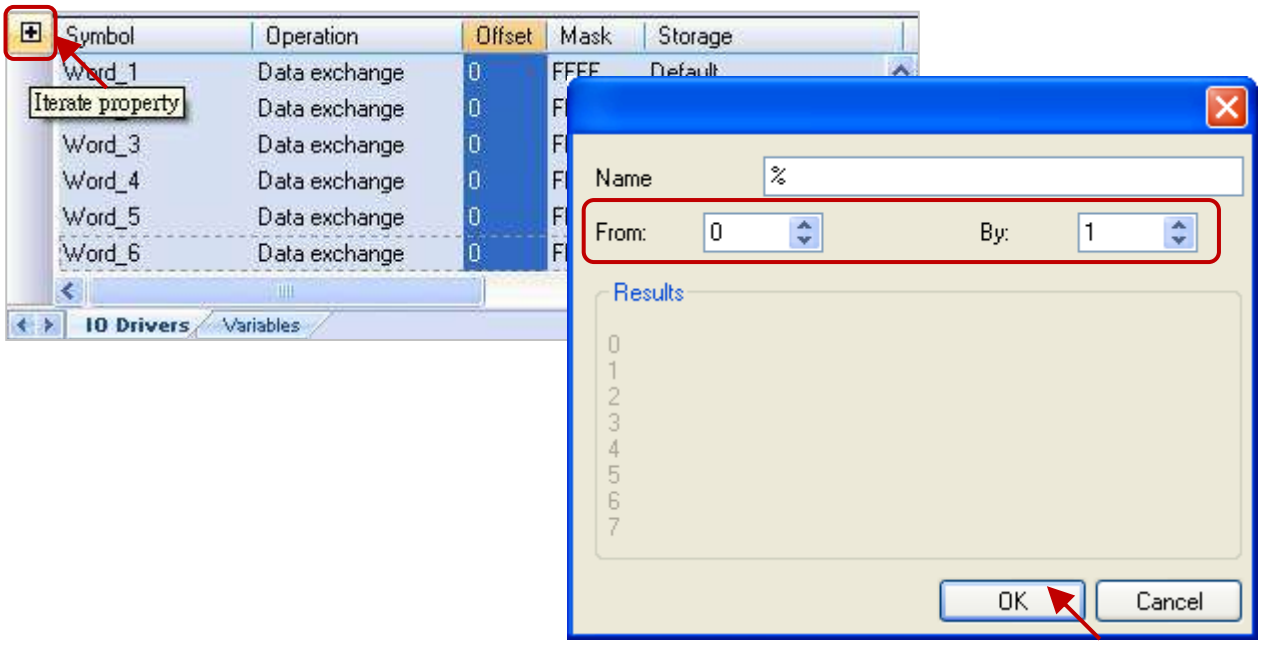
Refer the Appendix A for details on data type and ranges. After completing the settings, the defined variables show as below:



3. In the "IO Drivers" window, drag variables - "Word_1 to Word_6", "Long_1", "Real_1" and "Status" (that created in the Section 5.1.1) from the Variables Area to the Symbol Area in the third data block.
   Note: The "Status" is an array variable. When you drag "Status" into the Symbol Area, it will show "Status[0]" to "Status[4]", simply *press "Del" key to delete "Status[0] to [1]" and "Status[3] to [4]".*
4. Set the "Operation" field of the "Status[2]" as "Error report" (that means this variable will be set to an error code when a read error occurs, or reset it to "0" when a read request is successful).
   Press the "F1" key to see the description of the Modbus Master Configuration and move to the title "Status and command variables" to know related commands and error codes.
5. Both the "Long_1" and the "Real_1" are 32-bit variables and require two Modbus addresses. So, set their "Storage" column as "DWORD (Low – High)".



6. As the figure below, select the "Word_1" to “Word_6” and then click “Iterate property” to set their Offset value (From: 0 ; By: 1).

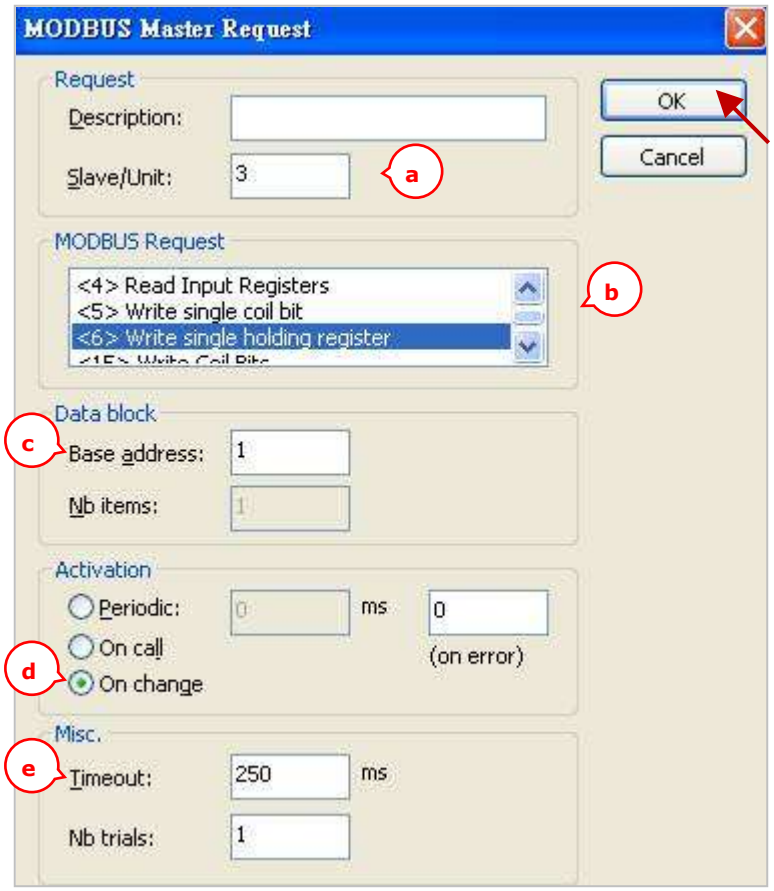


7. Next, double click the Offset field of “Long_1” and “Real_1” items and set their values as “6” and “8”, then press “Enter” key to complete the settings.

Note: One 32-bit data requires two Modbus addresses. For instance, the Offset value of “Long_1” is “6” and the next Offset value must be set to “8” (i.e., “Real_1”).

5.1.4 Write AO Data (16-bit)

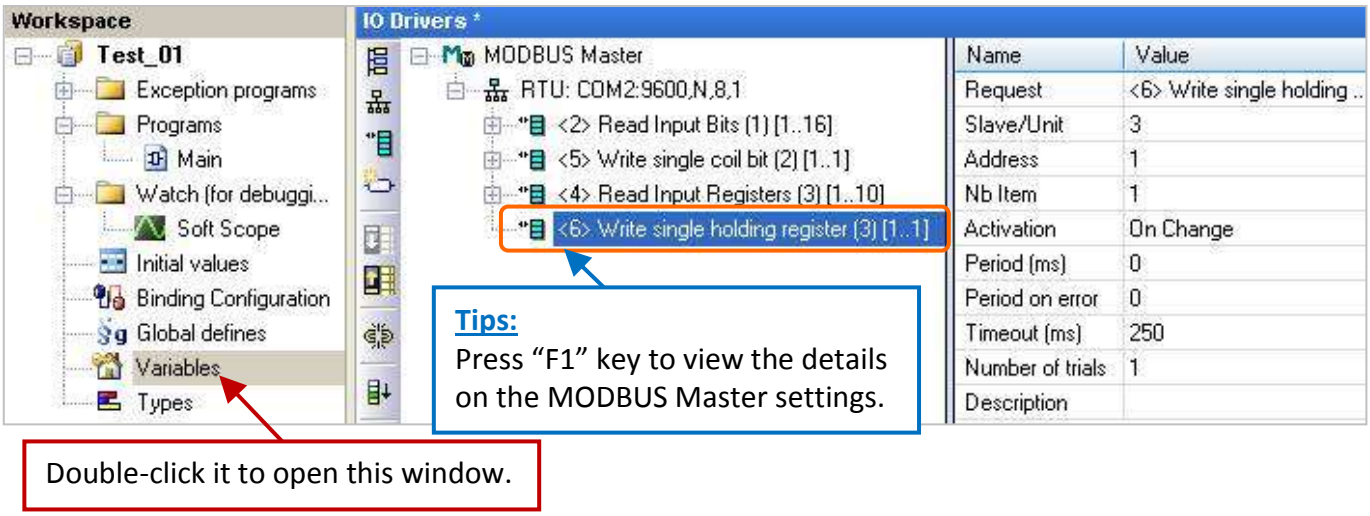
1. Using the same way in the Section 5.1 - Step 4 to create the 4th data block and completing all the following settings in the "MODBUS Master Request" window, and then click "OK".



In this example

a. Slave/Unit:
   Enter the Net-ID of the Slave device. (e.g., the Net-ID is "3").
b. MODBUS Request: Select "<6> Write single holding register".
c. Base address:
   Start from "1" by default.
   (Refer the Section 5.1.1 to change it.)
d. On change: In case of a write request, means that the request is activated each time any variable changed.
   (Refer the Section 5.1.1 for derails.)
e. Timeout: Set a timeout value.
   When time-out occurred, it will show the defined error code. (The recommended value for the Modbus RTU/ASCII device is 200 to 1000 ms. In this case the value is 250 ms.)

2. Next, open the "Variables" window and then declare variables that are available for the program.



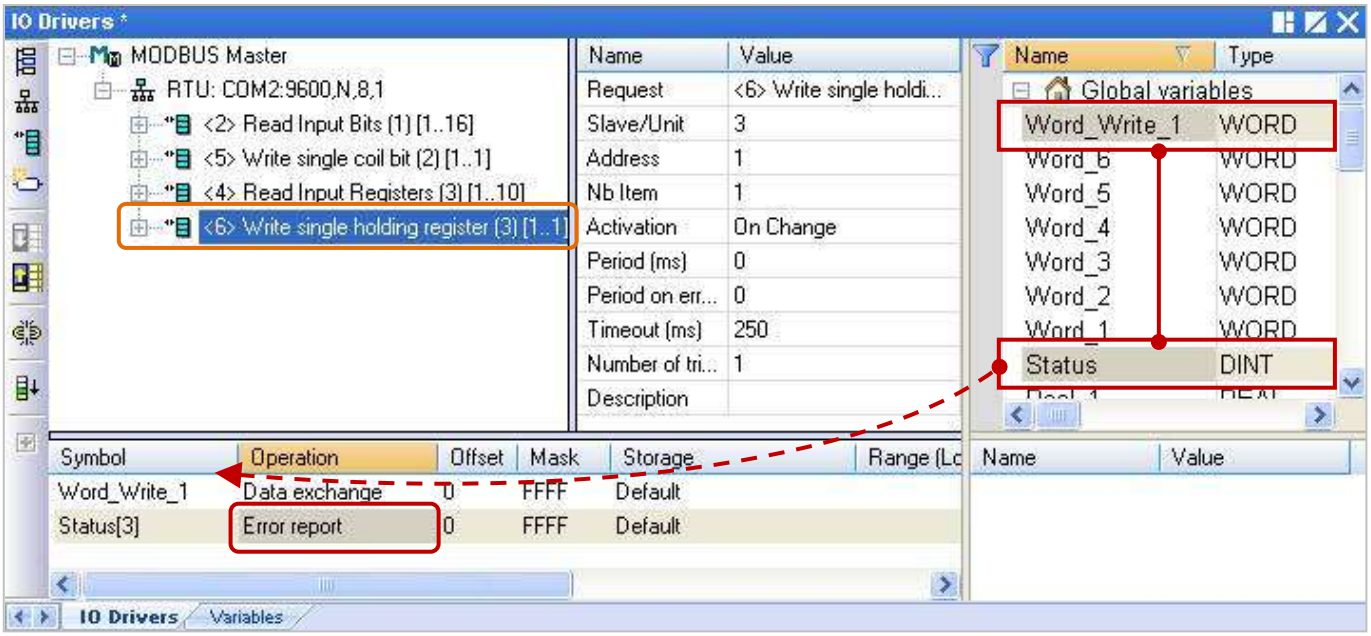
Declaring a "WORD" variable.
(Refer the Appendix A for details on data type and ranges ; refer the Section 2.3.1 for operations).

Variable name	Data type	Description
Word_Write_1	WORD	Used to write AO data (16-bit).

After completing the settings, the defined variables show as below:

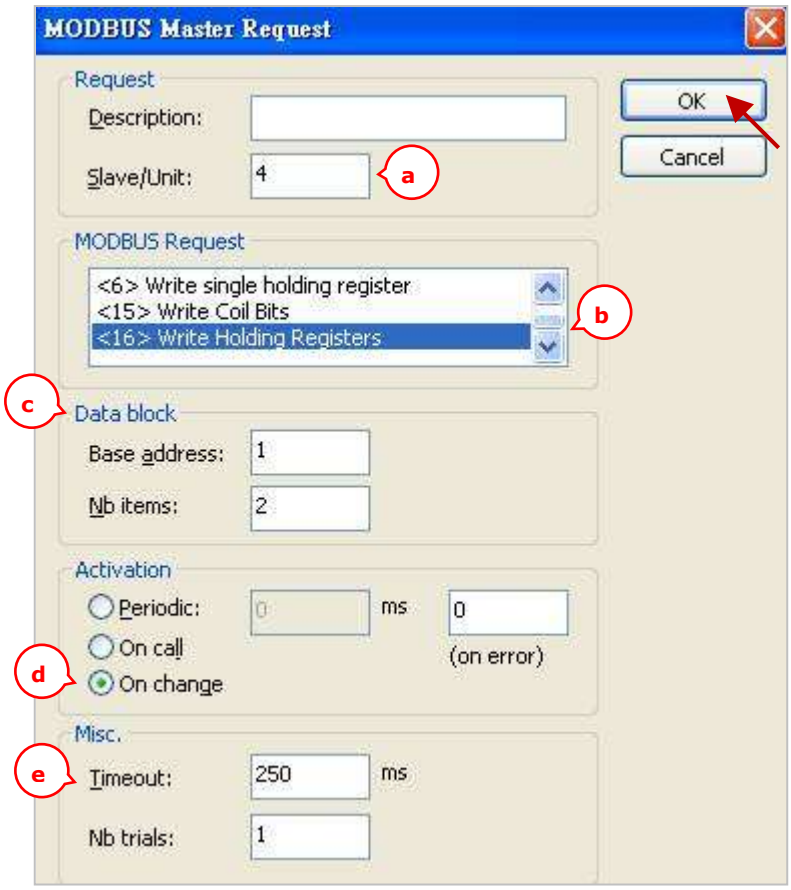
3. In the “IO Drivers” window, drag variables - “Word_Write_1” and “Status” (that created in the Section 5.1.1) from the Variables Area to the Symbol Area in the 4th data block.
   Note: The “Status” is an array variable. When you drag “Status” into the Symbol Area, it will show “Status[0]” to “Status[4]", *simply press “Del” key to delete “Status[0] to [2]” and “Status[4]".*

4. Set the “Operation” field of the “Status[3]” as “Error report” (that means this variable will be set to an error code when a read error occurs, or reset it to “0” when a read request is successful).
   Press the “F1” key to see the description of the Modbus Master Configuration and move to the title “Status and command variables” to know related commands and error codes.



5.1.5 Write AO Data (32-bit)

1. Using the same way in the Section 5.1 - Step 4 to create the 5th data block and completing all the following settings in the "MODBUS Master Request" window, and then click "OK".



In this example

a. Slave/Unit:
Enter the Net-ID of the Slave device. (e.g., the Net-ID is "4").

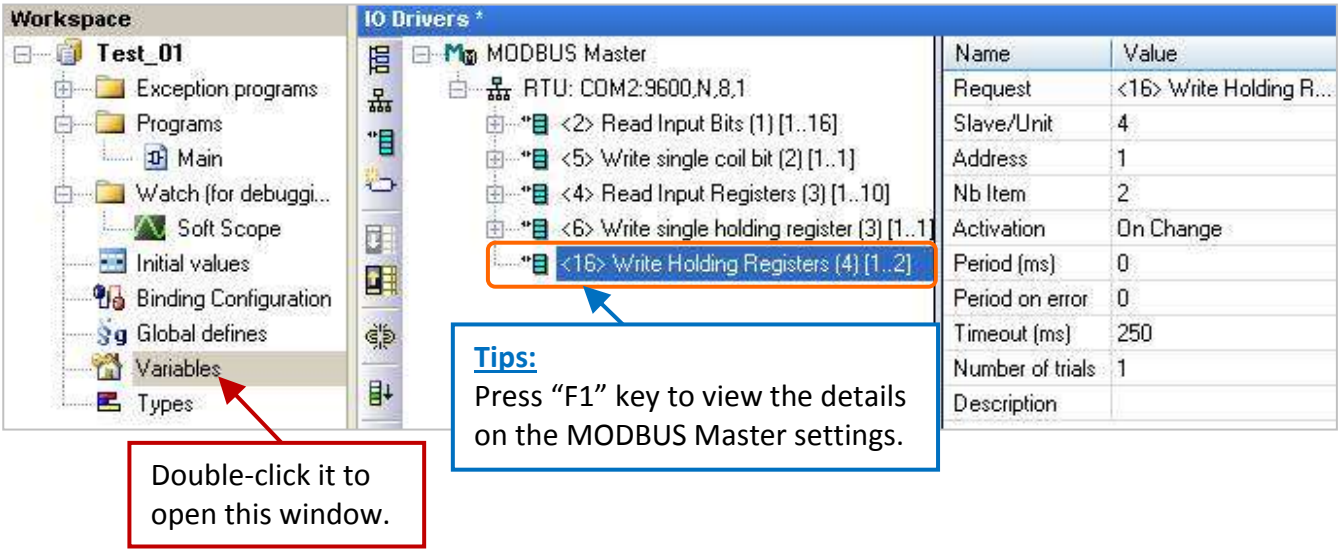
b. MODBUS Request:
Select "<16> Write Holding Registers".

c. Base address:
Start from "1" by default.
(Refer the Section 5.1.1 to change it.)
Nb items:
The number of AO signals to write.
(In this case, the number is "2" because the REAL type requires two Modbus address).

d. On change: In case of a write request, means that the request is activated each time any variable changed.
(Refer the Section 5.1.1 for details)

e. Timeout: Set a timeout value.
When time-out occurred, it will show the defined error code. (The recommended value for the Modbus RTU/ASCII device is 200 to 1000 ms. In this case the value is 250 ms.)

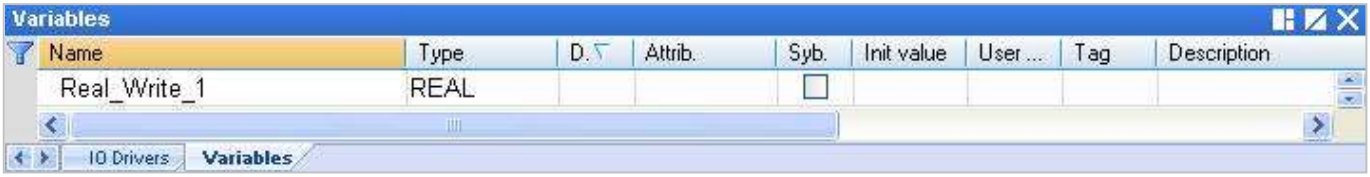
2. Next, open the "Variables" window and then declare variables that are available for the program.



Declaring a "Real" variable.
(Refer the Appendix A for details on data type and ranges ; refer the Section 2.3.1 for operations).

Variable name	Data type	Description
Real_Write_1	REAL	Used to write AO data (32-bit).

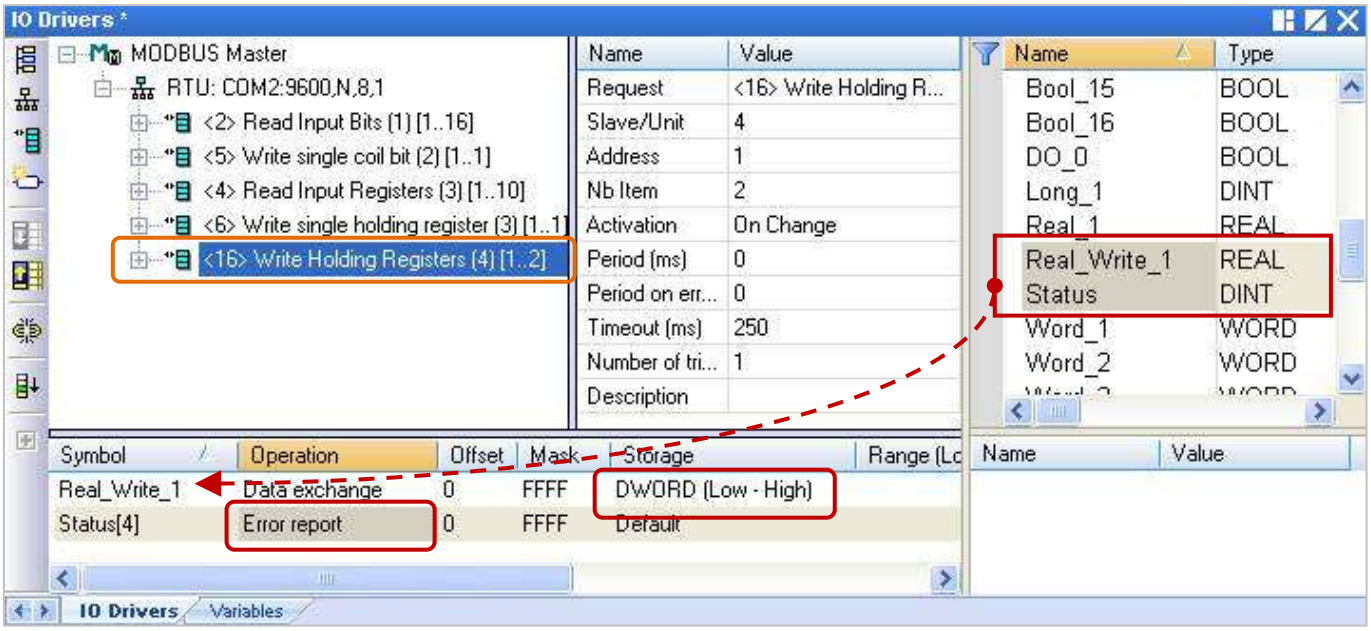
After completing the setting, the defined variable shows as below:



3. In the “IO Drivers” window, drag variables - “Real_Write_1” and “Status” (that created in the Section 5.1.1) from the Variables Area to the Symbol Area in the 5th data block.
   Note: The “Status” is an array variable. When you drag “Status” into the Symbol Area, it will show “Status[0]” to “Status[4]", simply *press “Del” key to delete “Status[0] to [3]”.*

4. Set the “Operation” field of the “Status[4]” as “Error report” (that means this variable will be set to an error code when a read error occurs, or reset it to “0” when a read request is successful).
   Press the “F1” key to see the description of the Modbus Master Configuration and move to the title “Status and command variables” to know related commands and error codes.

5. The “Real_Write_1” is a 32-bit data and required two Modbus addresses. So, set its “Storage” field as “DWORD (Low – High)”.



5.1.6 How to use the XV Board?

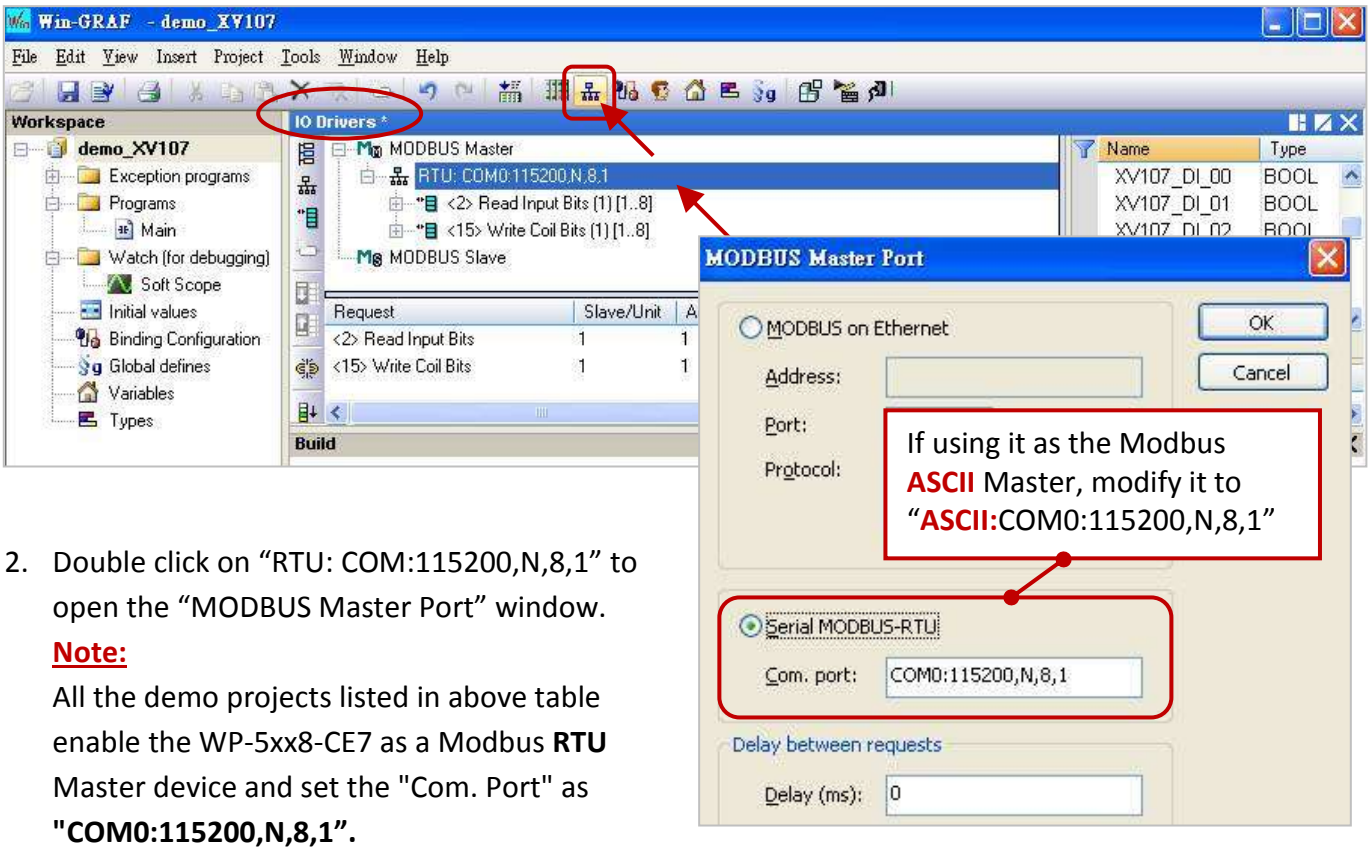
The XV board belongs to the Modbus Slave I/O board. Before using the I/O board, users must plug it into the WP-5xx8-CE7, and then enable the WP-5xx8-CE7 as a Modbus Master (refer the Section 5.1).
Visit the XV board Selection Guide page to get more details:
www.icpdas.com/root/product/solutions/hmi_touch_monitor/touchpad/xv-board_selection.html

All the Win-GRAF demo projects listed in the following table can be found on the CD-ROM. Refer the Chapter 12, click the Win-GARF menu bar "File" > "Add Existing Project" > "From Zip" to restore the demo project and to view the details. (CD-ROM:\Napdos\Win-GRAF\demo-project\)

Demo	File Name	Description
XV107, XV107A	demo_XV107.zip	Read 8 DI, Write 8 DO
XV110	demo_XV110.zip	Read 16 DI
XV111, XV111A	demo_XV111.zip	Read 16 DO, Read 1 DO
XV116	demo_XV116.zip	Read 5 DI, Write 6 D0
XV308_1 XV308_2 XV308_3	demo_XV308_1.zip demo_XV308_2.zip demo_XV308_3.zip	1. Read 8 AI, Read 8 DI 2. Read 8 AI, Write 8 DO 3. Read 8 AI, Write 4 DO, Read 4 DI
XV310	demo_XV310.zip	Read 4 AI, Write 4 DO, Read 4 DI, Write 4 AO

Common setting:

1. Mouse click the "Open Fieldbus Configuration" tool button to open the "I/O Drivers" window.



2. Double click on "RTU: COM:115200,N,8,1" to open the "MODBUS Master Port" window.

   Note:

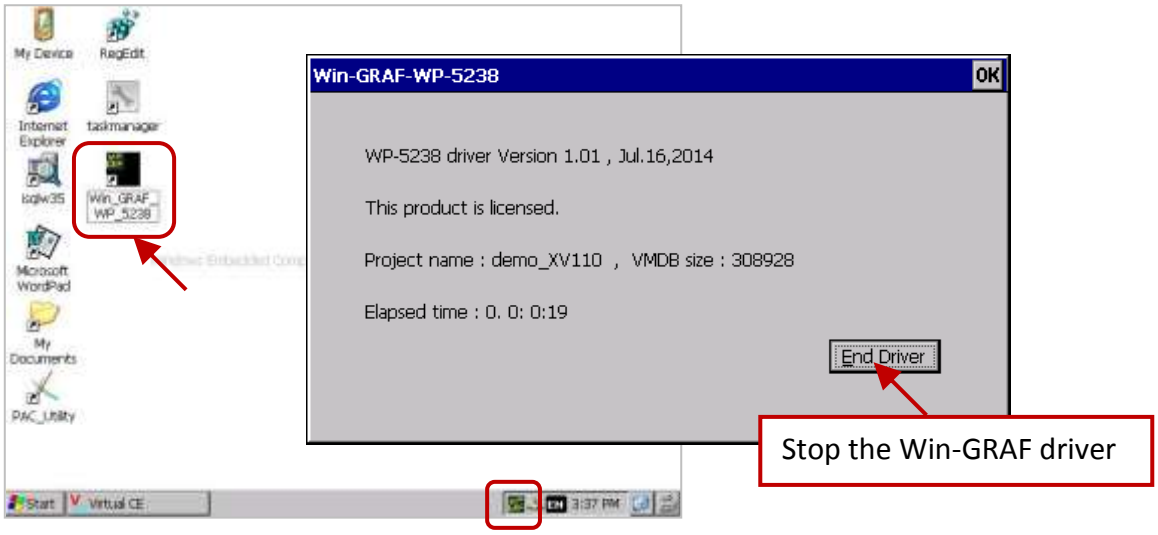
   All the demo projects listed in above table enable the WP-5xx8-CE7 as a Modbus **RTU** Master device and set the "Com. Port" as **"COM0:115200,N,8,1".**

Configure the AI/AO channel

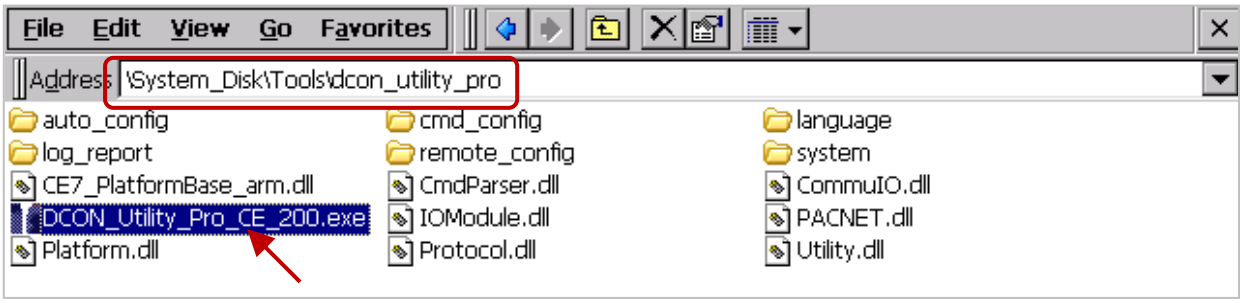
If you want to use the AI/AO channel of the XV Board (e.g., XV308, XV310) in the WP-5xx8-CE7. First, stop the Win-GRAF driver on the PAC and then configure each AI/AO channel by using "DCON_Utility_Pro_CE_200.exe".

Using the WP-5238-CE7 as an example:

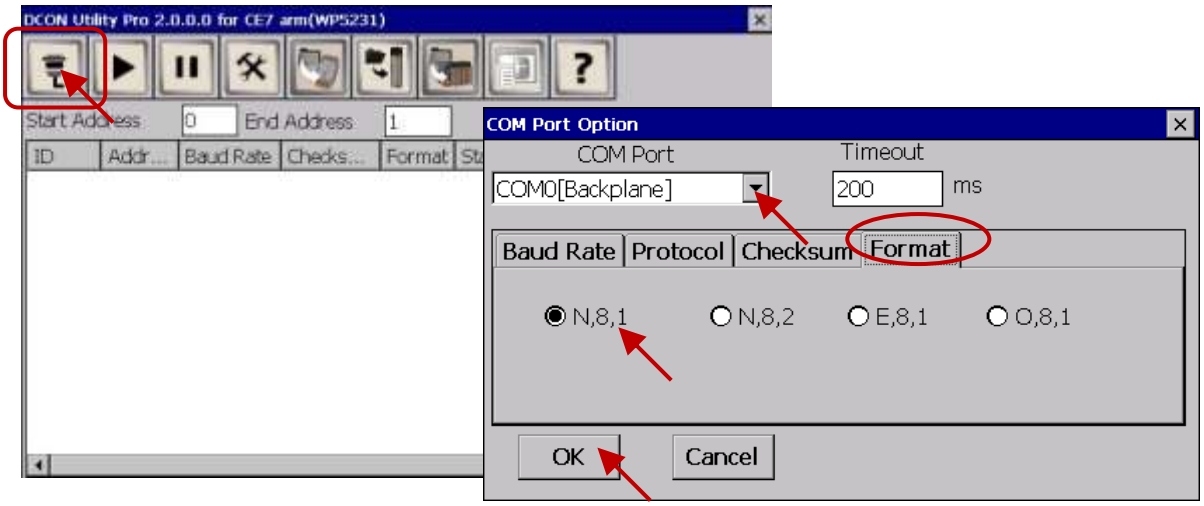
1. Click the "Win_GRAF_WP_5238" (or the small icon on the taskbar) to open the Win-GRAF driver window, and then click the "End Driver" button.



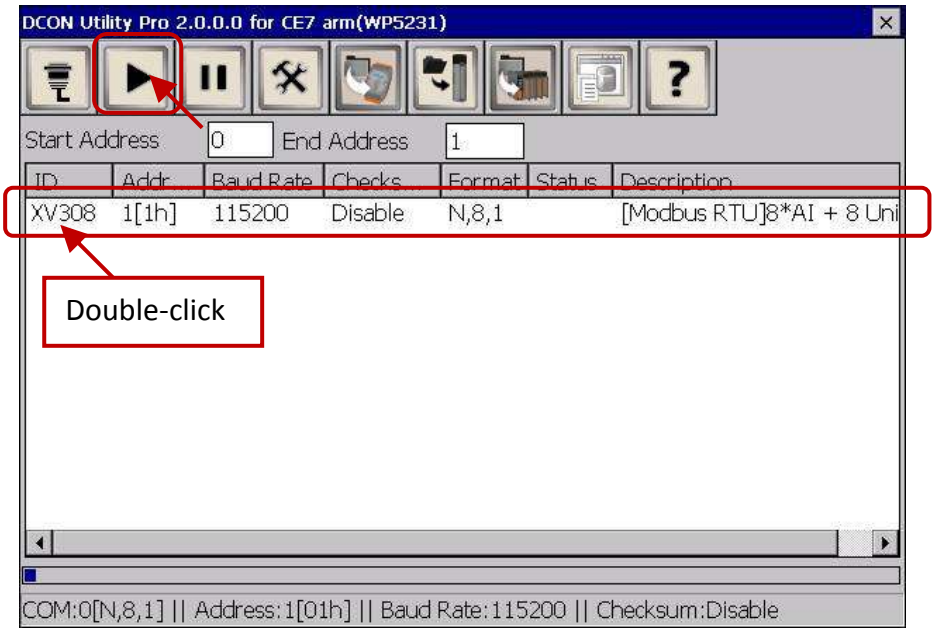
2. Click "My Device" on the desktop and then get into the path "\System_Disk\Tools\dcon_utility_pro" to run the "DCON_Utility_Pro_CE_200.exe".



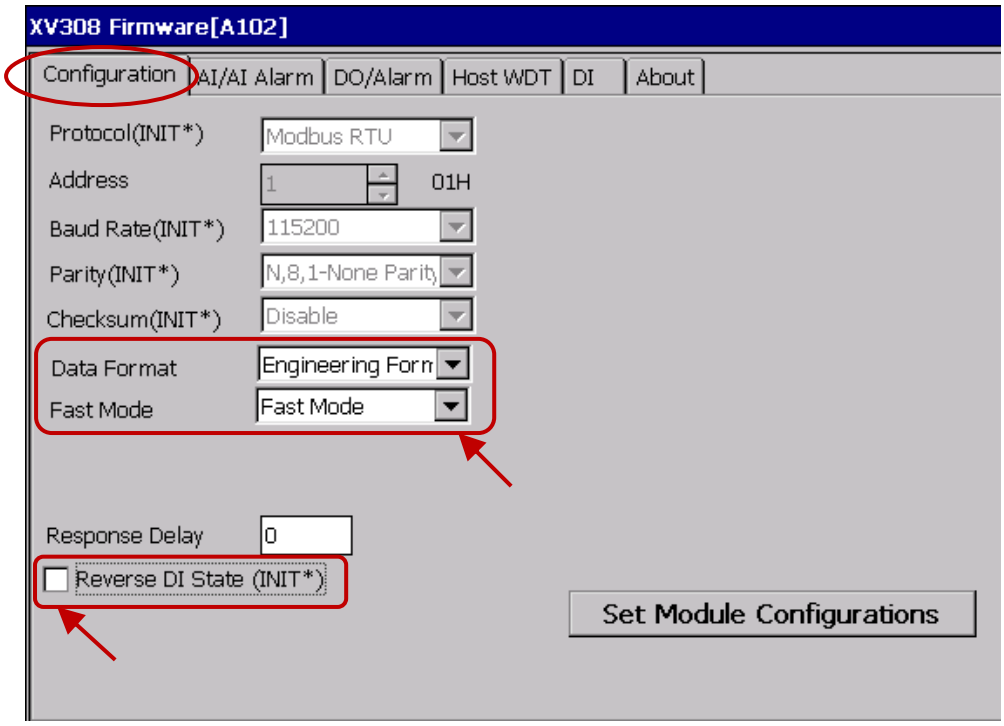
3. Click the COM Port button to set the "COM Port" as "COM0", set the "Baud Rate" as "115200" and set the "Format" as "N,8,1", and then click "OK".



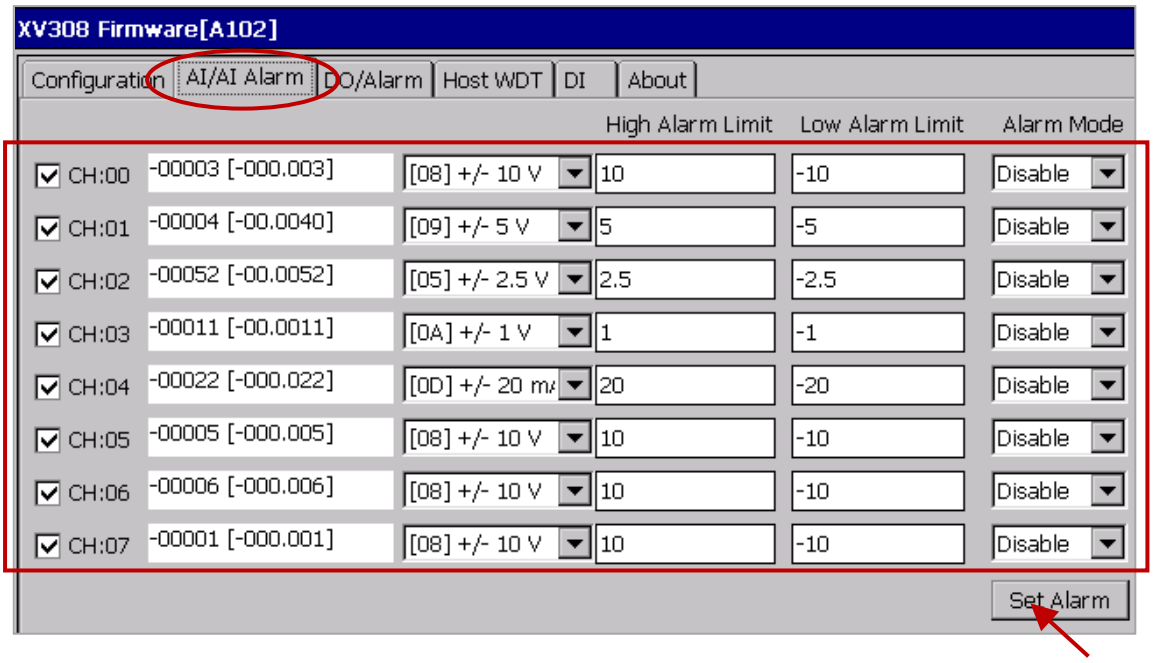
4. After clicking the Search button, the XV Board (e.g., XV308) will show in the window. Then, double click this item to get into the setting window.



5. In the "Configuration" tab, set the "Data Format" as "Engineering Format" (recommended setting), set the "Fast Mode" as "Fast Mode" and uncheck the "Reverse DI State (INIT*)".



6. In the "AI/AI Alarm" tab, to configure the proper ranges and values for each AI channel, and remenmber to select any AI channel (e.g.,"CH:00") you want to use, then click the "Set Alarm" button.



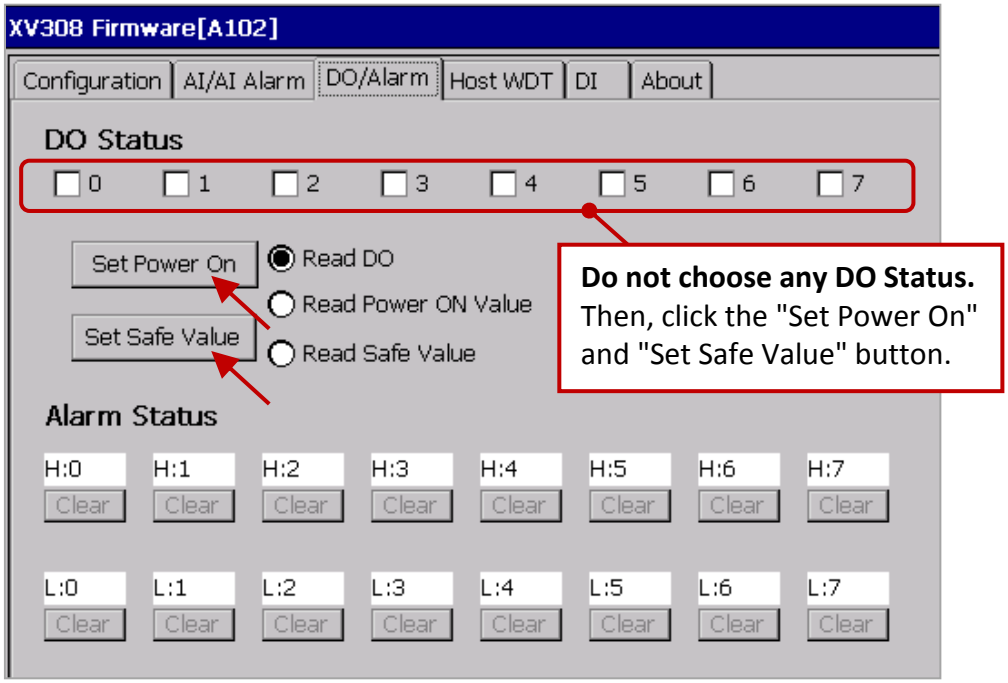
XV308:

Type Code	Range	Data Format	Minimum	Maximum
05	+/-2.5 V	Engineering	-25000	+25000
		Hexadecimal	8000h	7FFFh
06	+/-20 mA	Engineering	-20000	+20000
		Hexadecimal	8000h	7FFFh
07	+4 mA ~ +20 mA	Engineering	+4000	+20000
		Hexadecimal	0000h	FFFFh
08	+/-10 V	Engineering	-10000	+10000
		Hexadecimal	8000h	7FFFh
09	+/-5 V	Engineering	-5000	+5000
		Hexadecimal	8000h	7FFFh
0A	+/-1 V	Engineering	-10000	+10000
		Hexadecimal	8000h	7FFFh
0D	+/-20 mA	Engineering	-20000	+20000
		Hexadecimal	8000h	7FFFh
1A	0 mA ~ +20 mA	Engineering	0	+20000
		Hexadecimal	0000h	FFFFh

Note:

1. For easy to use, recommended to use the data format - "Engineering". (E.g., "+/-2.5 V" will show as "-25000 to +25000" and "+4 mA to +20 mA" will show as "+4000 to +20000")
2. When using these "Type Code" - 06, 07, 0D, 1A, please check if the position of eight hardware jumpers on the XW board are correct.
   www.icpdas.com/root/product/solutions/datasheet/hmi_touch_monitor/XV308.pdf

Note: When using the XV308, you need to click the "Set Power On" and "Set Safe Value" button (do not choose any DO Status) in the "DO/Alarm" tab.



7. Finally, back to the "Configuration" tab and click the "Set Module Configuration" button (Step5) to finish the AI/AO configuration, and then close the "DCON_Utility_Pro_CE_200.exe". In addition, click the "Win_GRAF_WP_5238" on the desktop to run the Win-GRAF driver (like Step 1).

Follow the similar way like the steps above to configure the AI/AO of the XV310.

XV310 - Analog Input:

Type Code	Range	Data Format	Minimum	Maximum
05	+/-2.5 V	Engineering	-25000	+25000
		Hexadecimal	8000h	7FFFh
06	+/-20 mA	Engineering	-20000	+20000
		Hexadecimal	8000h	7FFFh
07	+4 mA ~ +20 mA	Engineering	+4000	+20000
		Hexadecimal	0000h	FFFFh
08	+/-10 V	Engineering	-10000	+10000
		Hexadecimal	8000h	7FFFh
09	+/-5 V	Engineering	-5000	+5000
		Hexadecimal	8000h	7FFFh
0A	+/-1 V	Engineering	-10000	+10000
		Hexadecimal	8000h	7FFFh
0D	+/-20 mA	Engineering	-20000	+20000
		Hexadecimal	8000h	7FFFh
1A	0 mA ~ +20 mA	Engineering	0	+20000
		Hexadecimal	0000h	FFFFh

Note:

1. For easy to use, recommended to use the data format - "Engineering". (E.g., "+/-2.5 V" will show as "-25000 to +25000" and "+4 mA to +20 mA" will show as "+4000 to +20000")
2. When using these "Type Code" - 0, 1, 06, 07, 0D, 1A, please check if the position of eight hardware jumpers on the XW board are correct.
   www.icpdas.com/root/product/solutions/datasheet/hmi_touch_monitor/XV310.pdf

XV310 - Analog Output:

Type Code	Range	Data Format	Minimum	Maximum
0	0 mA ~ +20 mA	Engineering	0	+20000
		Hexadecimal	0000h	FFFFh
1	+4 mA ~+20 mA	Engineering	+4000	+20000
		Hexadecimal	0000h	FFFFh
2	0V ~ +10 V	Engineering	0	+10000
		Hexadecimal	0000h	FFFFh
3	+/-10 V	Engineering	-10000	+10000
		Hexadecimal	8000h	7FFFh
4	0 V ~ +5 V	Engineering	0	+5000
		Hexadecimal	0000h	FFFFh
5	+/-5 V	Engineering	-5000	+5000
		Hexadecimal	8000h	7FFFh

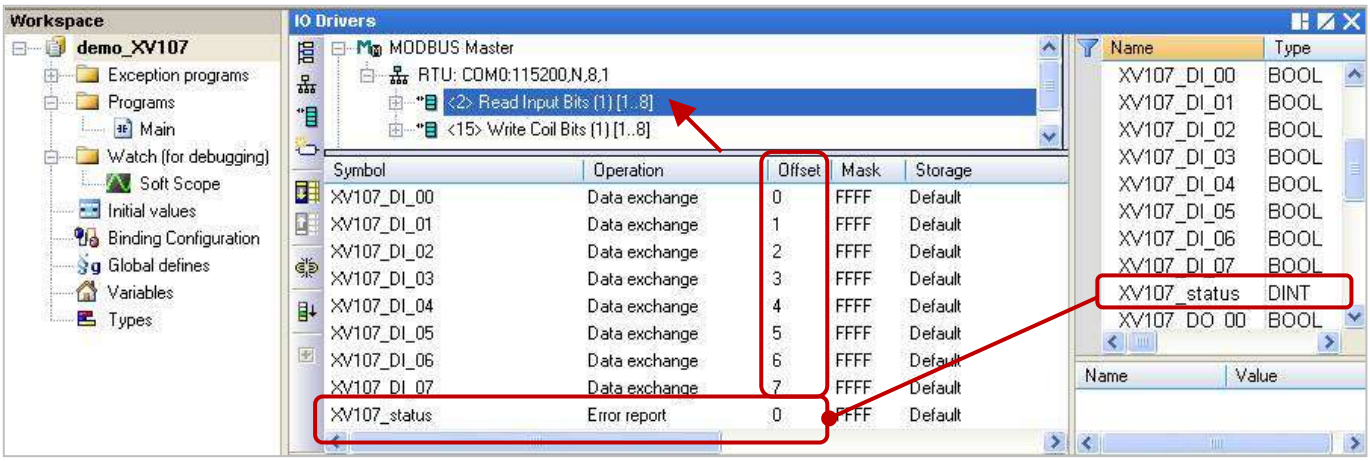
5.1.7 Connecting the XV107/ XV107A (8 DI, 8 DO)

The XV107/XV107A is an 8-channel digital input and 8-channel digital output board. This section provides a Win-GRAF demo project - "demo_XV107.zip". First, go to Section 5.1.6 for the information of the XV Board before using it.

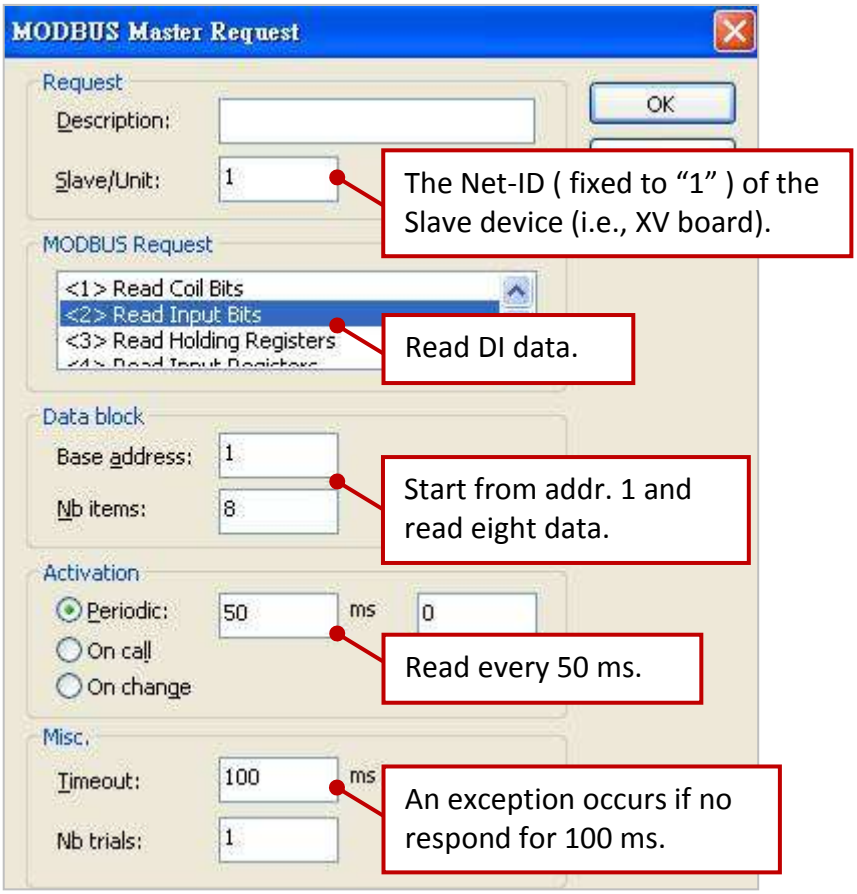
Demo description:

This demo added two data blocks. One is used to read 8 DI data and the other is used to write 8 DO data.

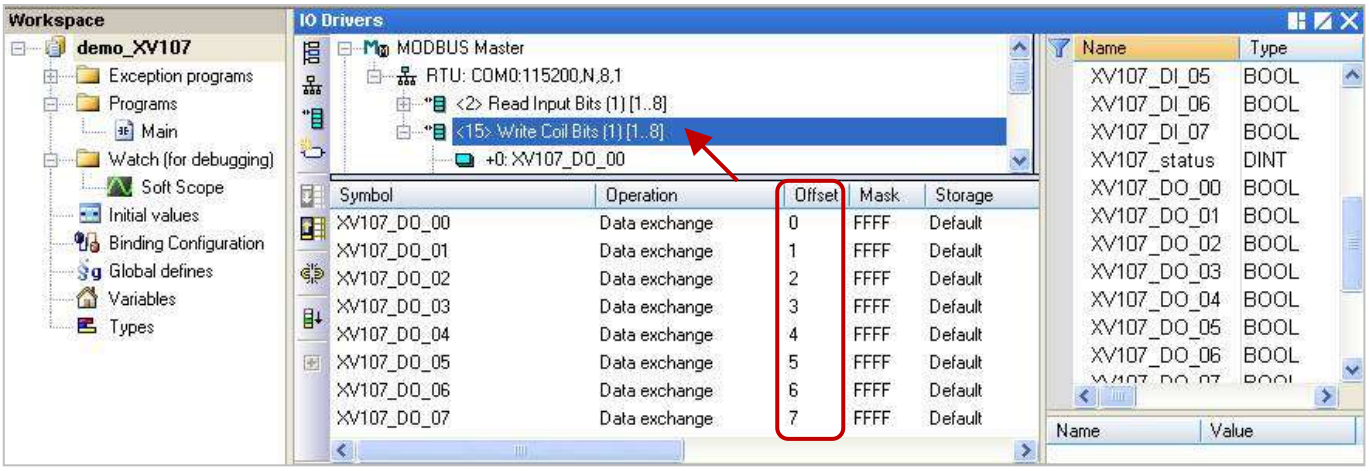
1. Mouse double click the 1st data block (i.e., <2> Read Input Bits) to open the setting window.

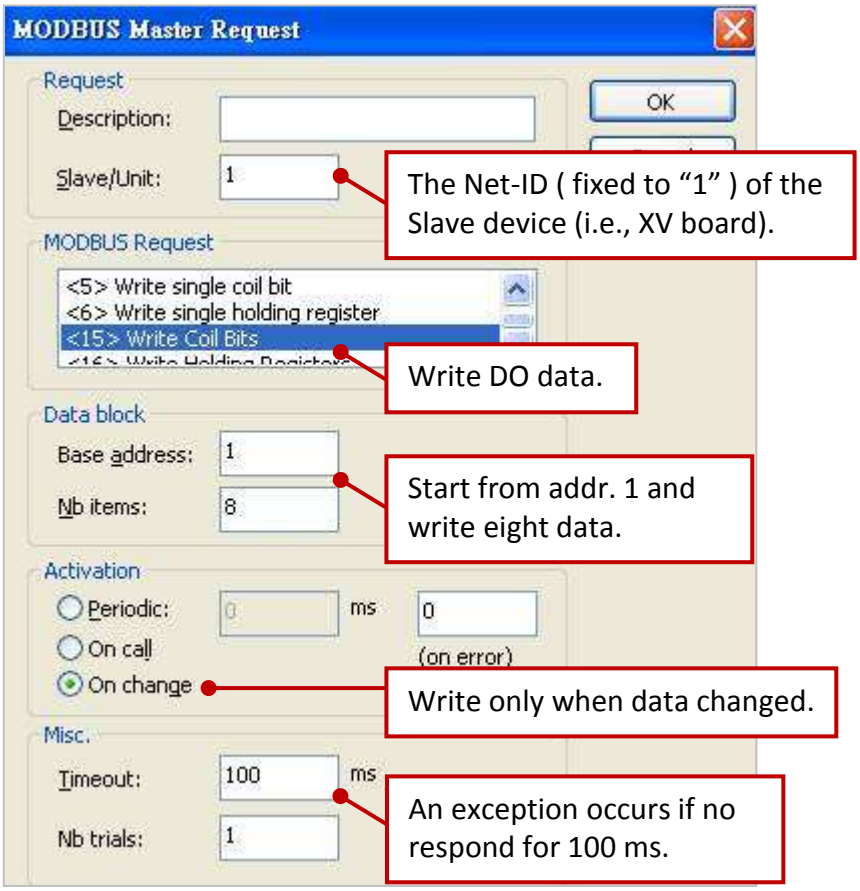


Note: The "Offset" value starts at "0" and the Modbus address of variable is equal to the "Offset" value plus 1 (Base address). Moreover, if you set the "Operation" as "Error report", the "Offset" value for the mapping variable (Date Type: DINT) must set to "0".



2. Mouse double click the 2nd data block (i.e., <15> Write Coil Bits) to open the setting window.





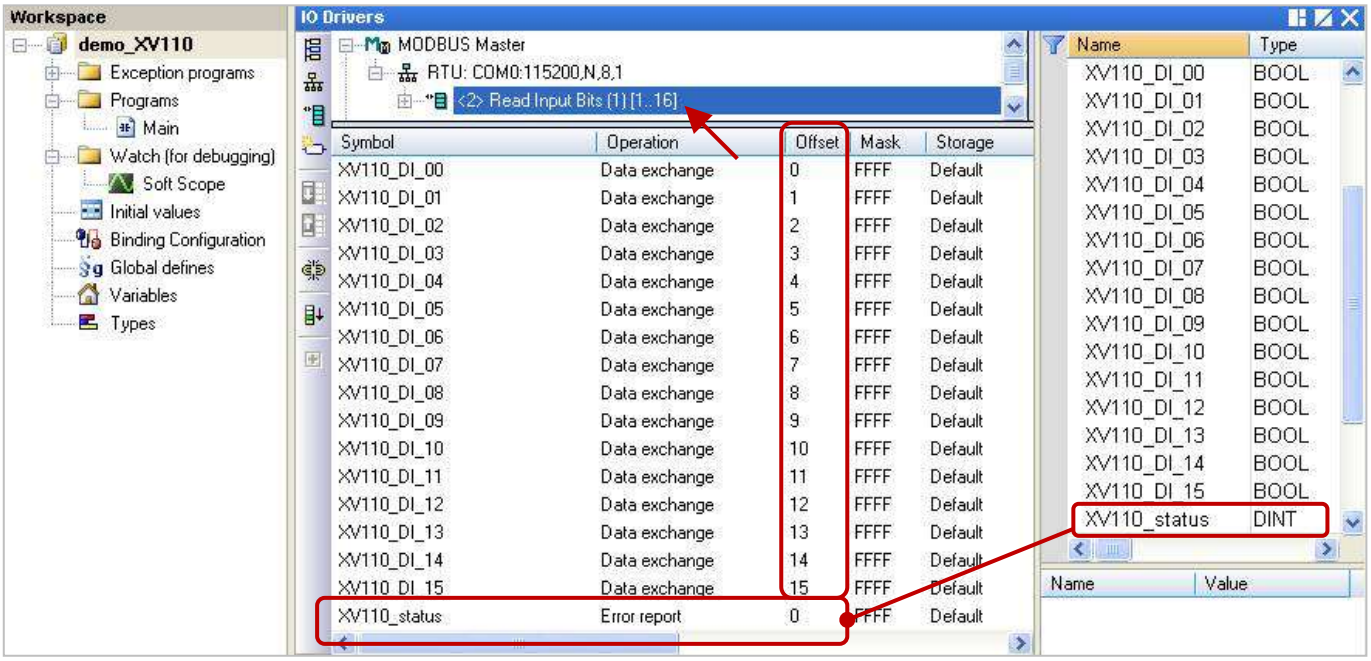
5.1.8 Connecting the XV110 (16 DI)

The XV110 is a 16-channel digital input board. This section provides a Win-GRAF demo project - "demo_XV110.zip". First, go to Section 5.1.6 for the information of the XV Board before using it.

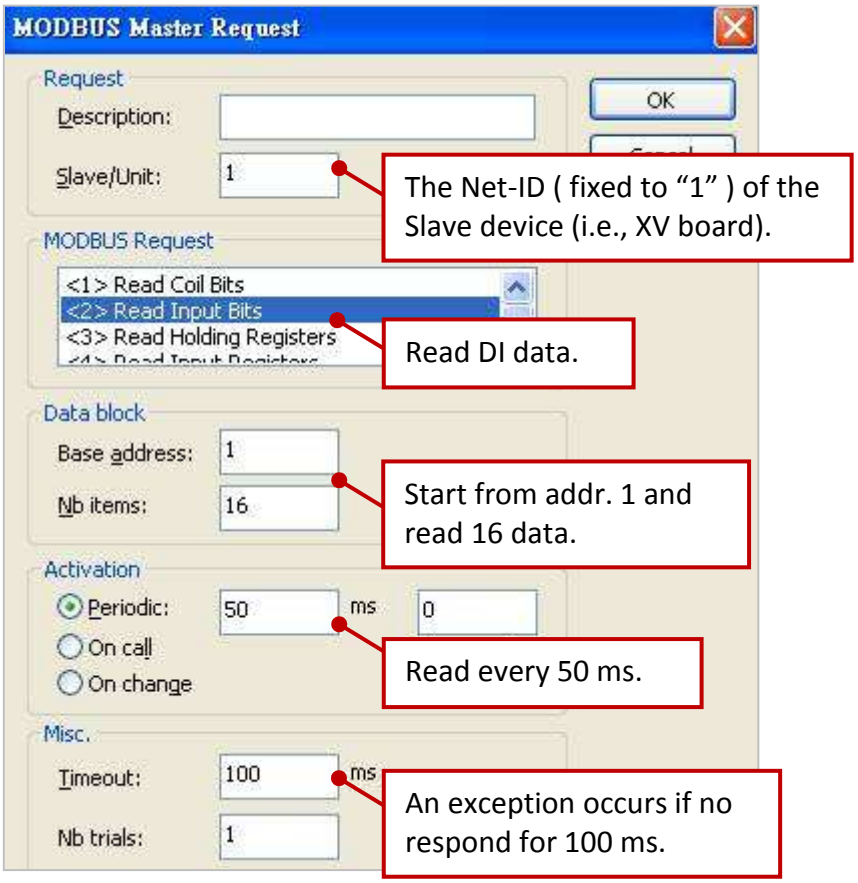
Demo description:

This demo added one data block that used to write 16 DI data.

1. Mouse double click "<2> Read Input Bits" to open the setting window.



Note: The "Offset" value starts at "0" and the Modbus address of variable is equal to the "Offset" value plus 1 (Base address). Moreover, if you set the "Operation" as "Error report", the "Offset" value for the mapping variable (Date Type: DINT) must set to "0".



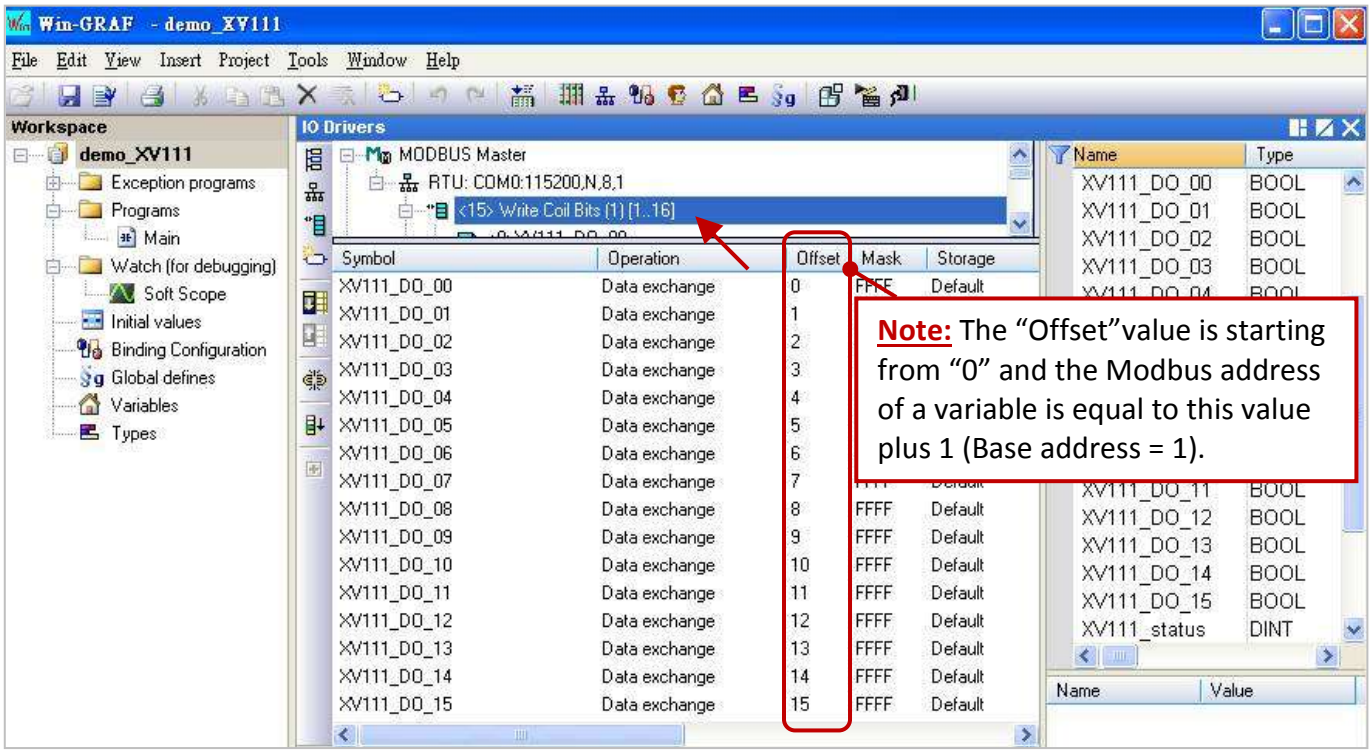
5.1.9 Connecting the XV111, XV111A (16 DO)

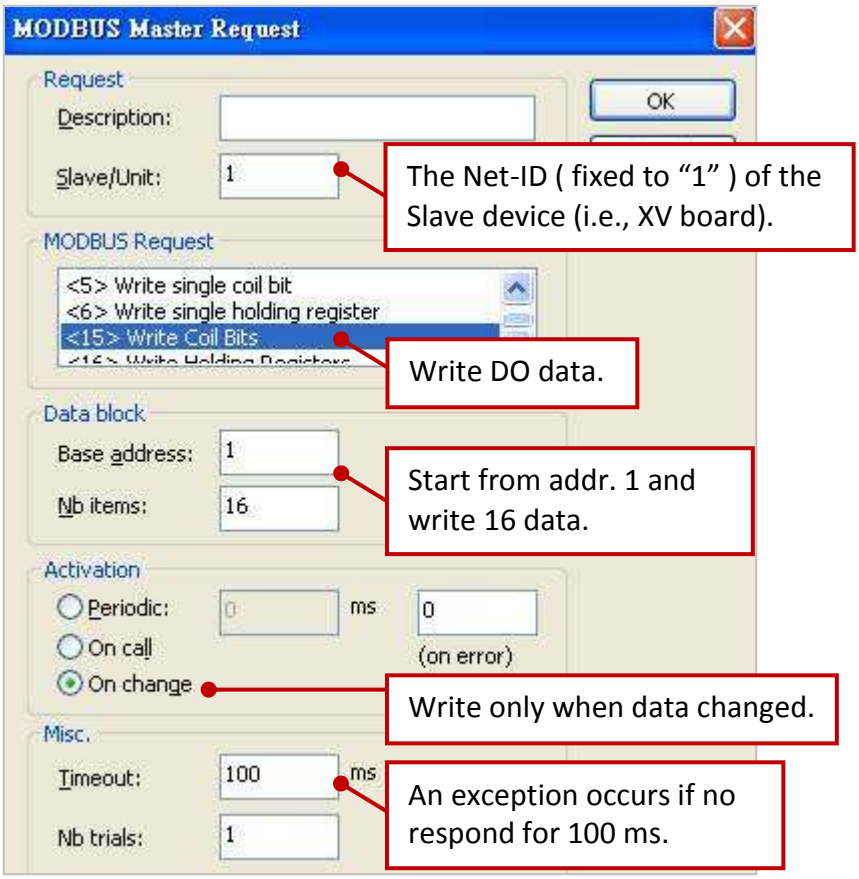
The XV111/ XV111A is a 16-channel digital output board. This section provides a Win-GRAF demo project - "demo_XV111.zip". First, go to Section 5.1.6 for the information of the XV Board before using it.

Demo description:

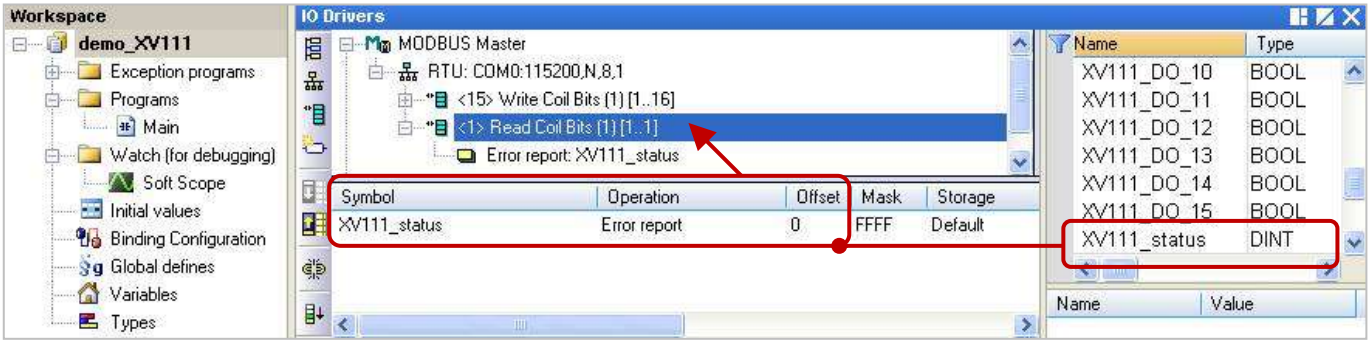
This demo added two data blocks. One is used to write 16 DO data and the other is used to read the DO status.

1. Mouse double click the 1st data block (i.e., <15> Write Coil Bits) to open the setting window.

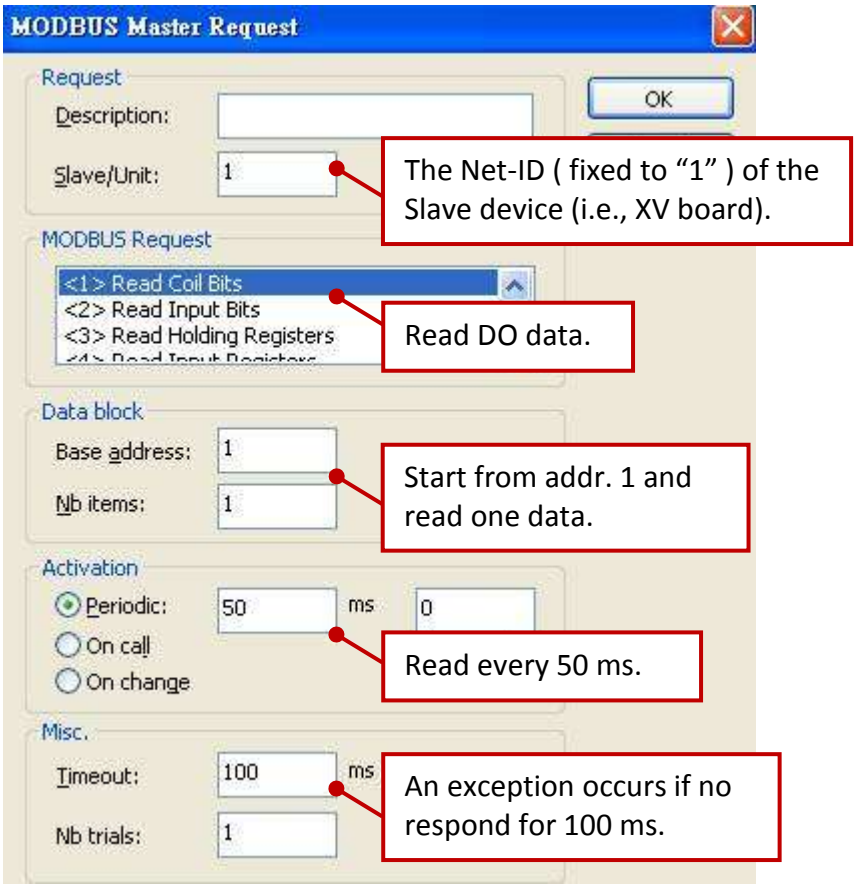




2. Mouse double click the 2nd data block (i.e., <1> Read Coil Bits) to open the setting window.



Note: The "Offset" value starts at "0" and the Modbus address of variable is equal to the "Offset" value plus 1 (Base address).



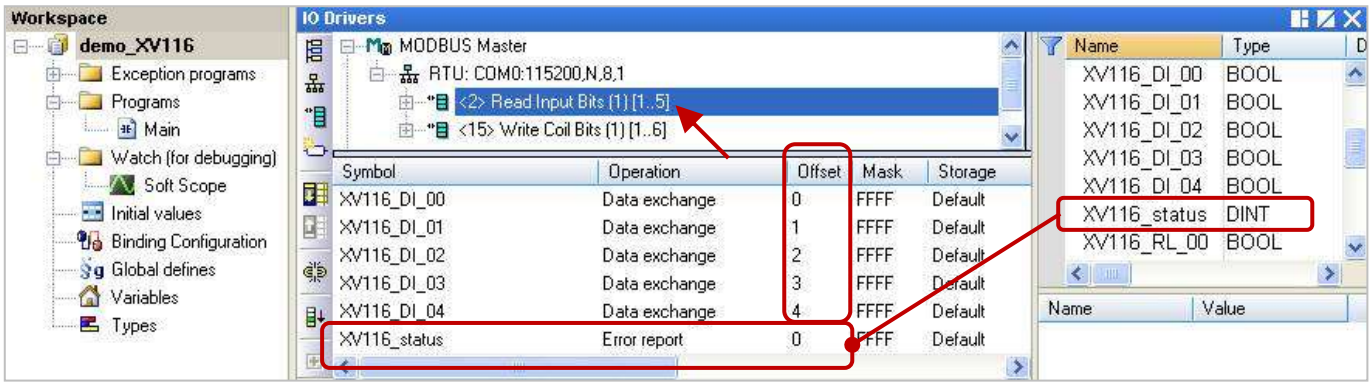
5.1.10 Connecting the XV116 (5 DI, 6 Relay)

The XV116 is a 5-channel digital input and 6-channel relay output board. This section provides a Win-GRAF demo project - "demo_XV116.zip". First, go to Section 5.1.6 for the information of the XV Board before using it.

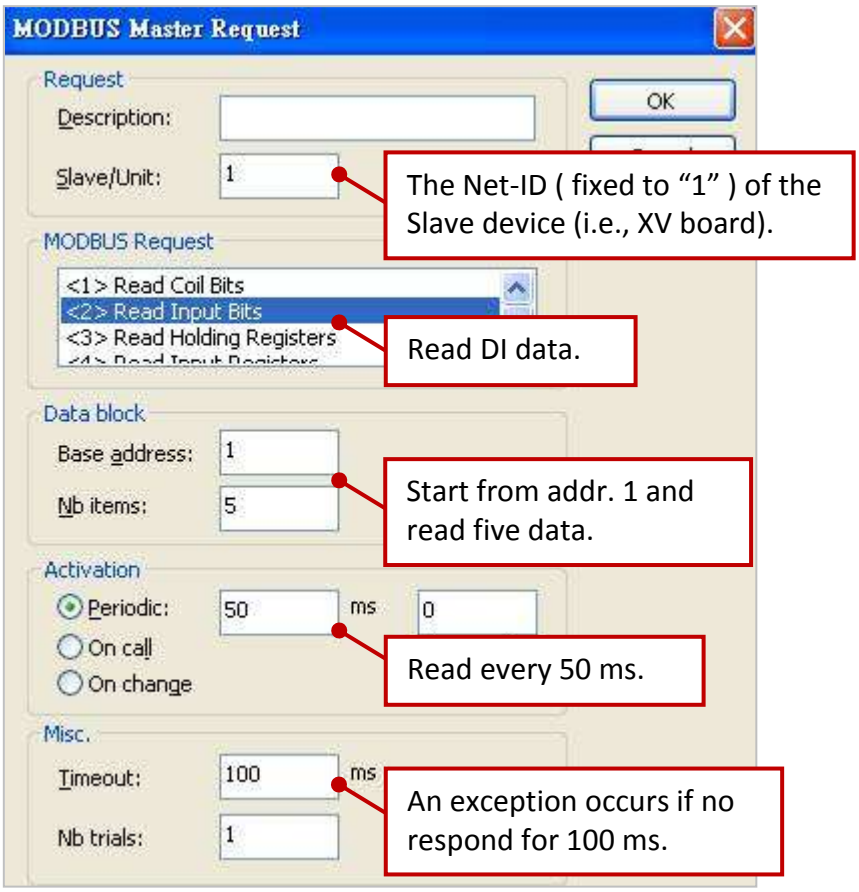
Demo description:

This demo added two data blocks. One is used to read 5 DI data and the other is used to write 6 DO data.

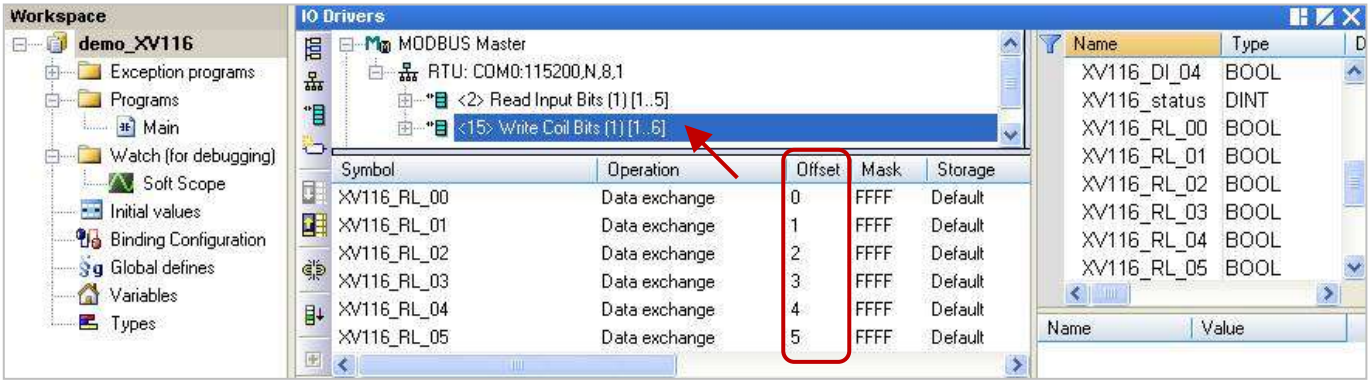
1. Mouse double click the 1st data block (i.e., <2> Read Input Bits) to open the setting window.

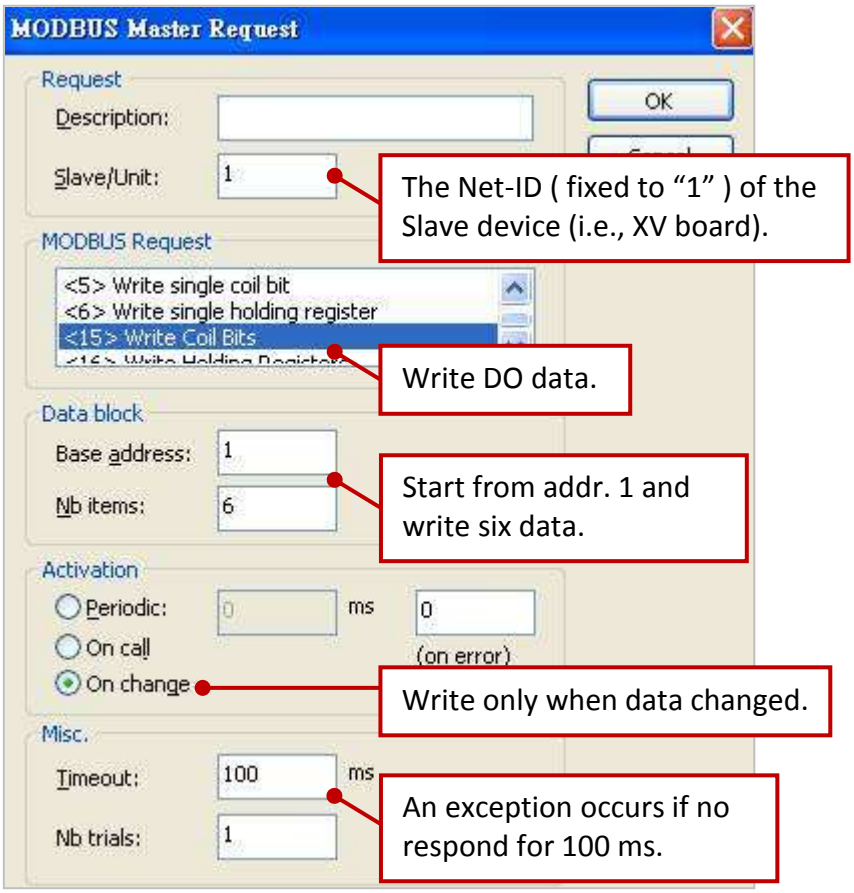


Note: The "Offset" value starts at "0" and the Modbus address of variable is equal to the "Offset" value plus 1 (Base address). Moreover, if you set the "Operation" as "Error report", the "Offset" value for the mapping variable (Date Type: DINT) must set to "0".



2. Mouse double click the 2nd data block (i.e., <15> Write Coil Bits) to open the setting window.





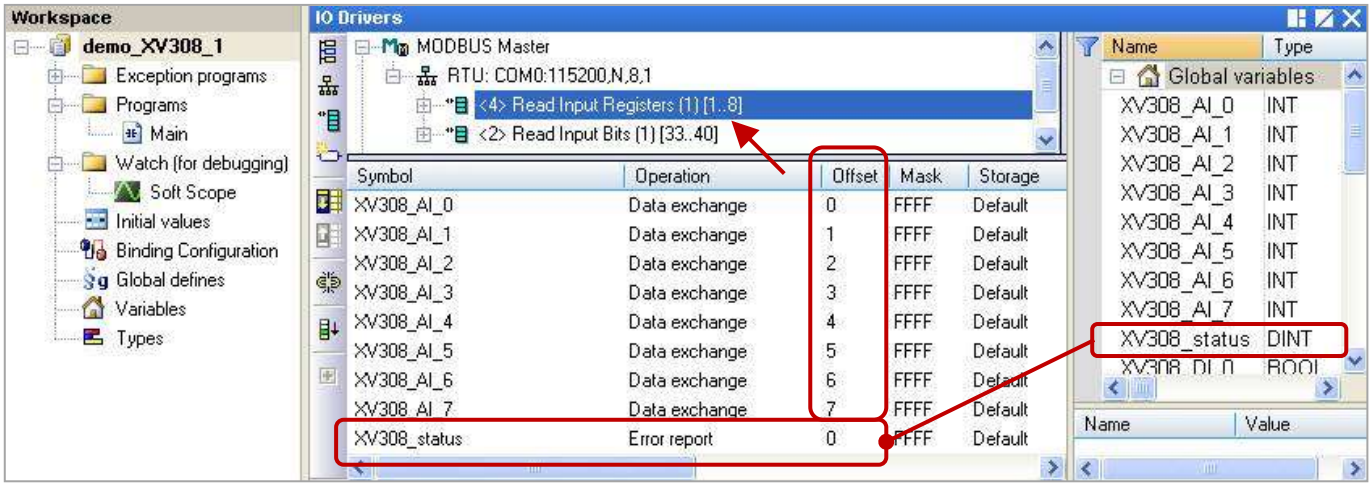
5.1.11 Connecting the XV308 (8 AI, 8 DIO)

The XV308 is a 8-channel analog input and 8-channel digital input/output board. This section provides three Win-GRAF demo projects - "demo_XV308_1.zip", "demo_XV308_2.zip" and "demo_XV308_3.zip". First, go to Section 5.1.6 to view the XV Board instructions and then configure each AI channel by using "DCON_Utility_Pro_CE_200.exe".

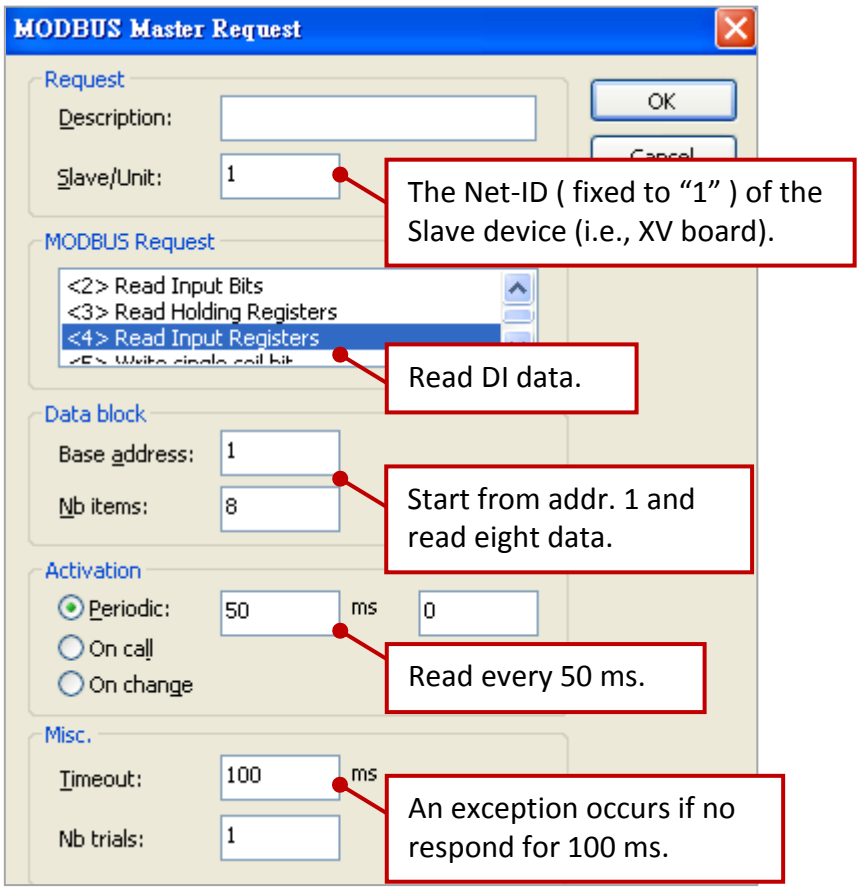
Demo description: (demo_XV308_1)

This demo added two data blocks, one is used to read 8 AI data and the other is used to read 8 DI data.

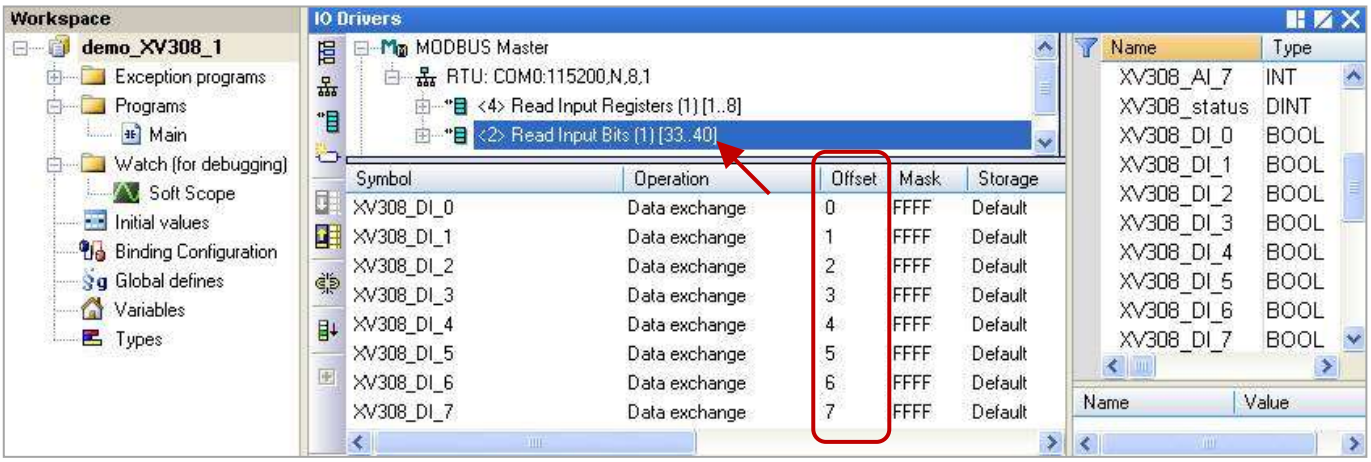
1. Mouse double click the 1st data block (i.e., <4> Read Input Registers) to open the setting window.



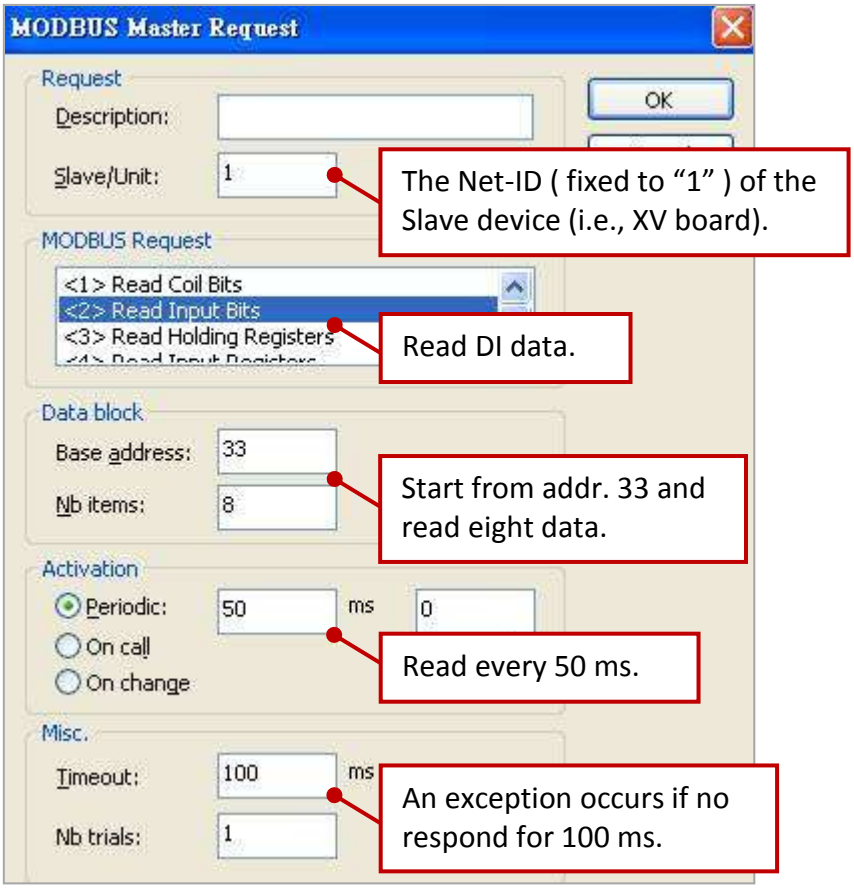
Note: The "Offset" value starts at "0" and the Modbus address of variable is equal to the "Offset" value plus 1 (Base address). Moreover, if you set the "Operation" as "Error report", the "Offset" value for the mapping variable (Date Type: DINT) must set to "0".



2. Mouse double click the 2nd data block (i.e., <2> Read Input Bits) to open the setting window.



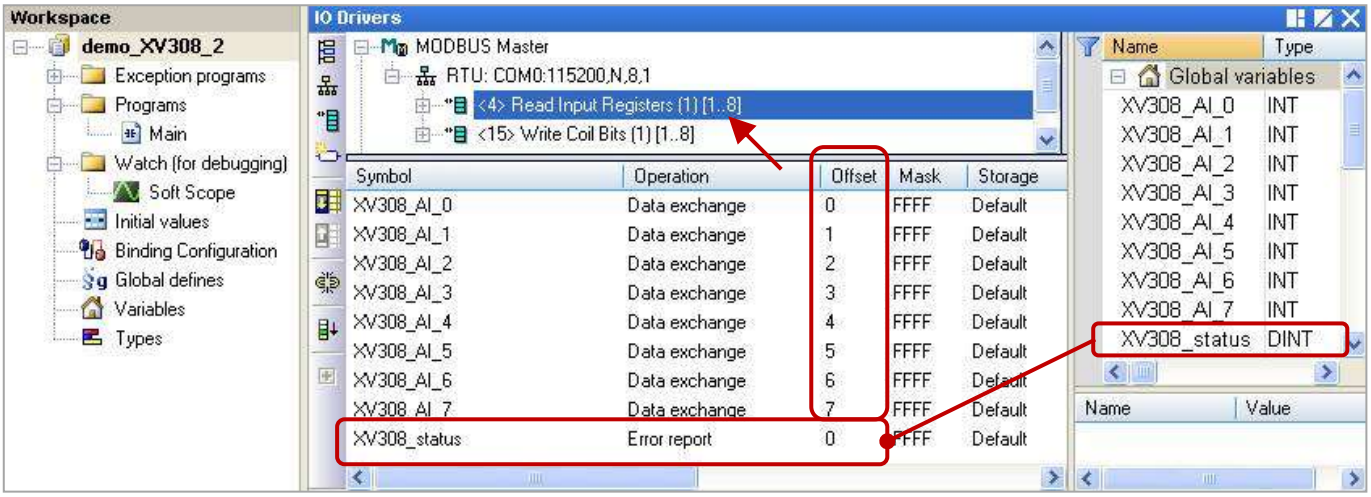
Notw: When using the XV308 to read DI data, the address must start from "33".



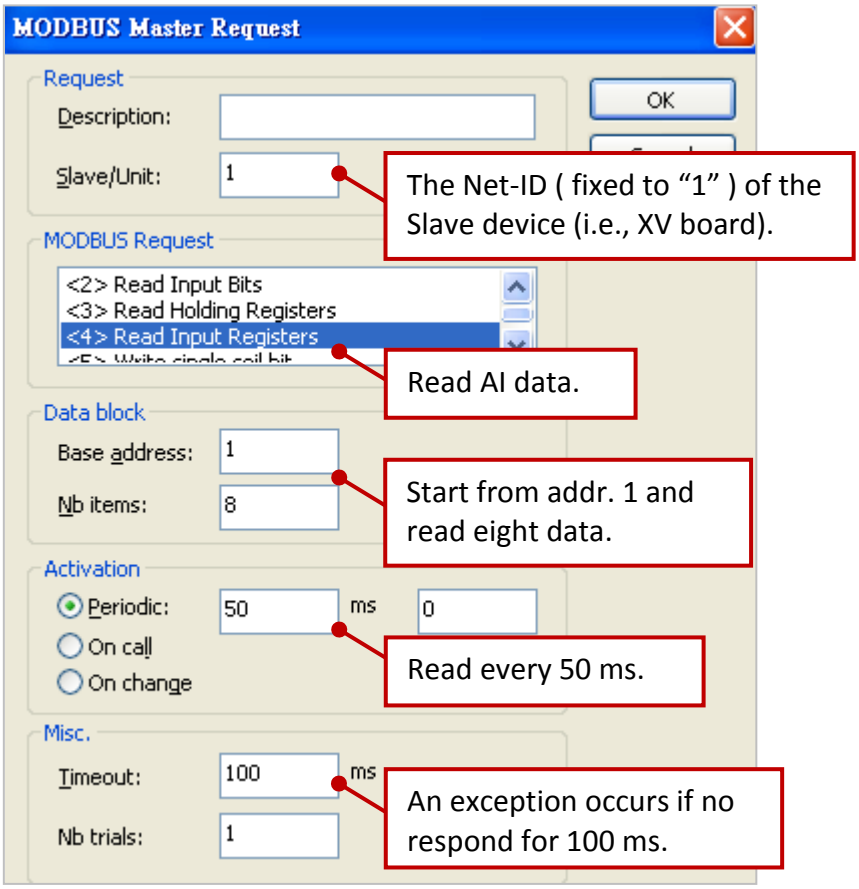
Demo description: (demo_XV308_2)

This demo added two data blocks, one is used to read 8 AI data and the other is used to write 8 DO data.

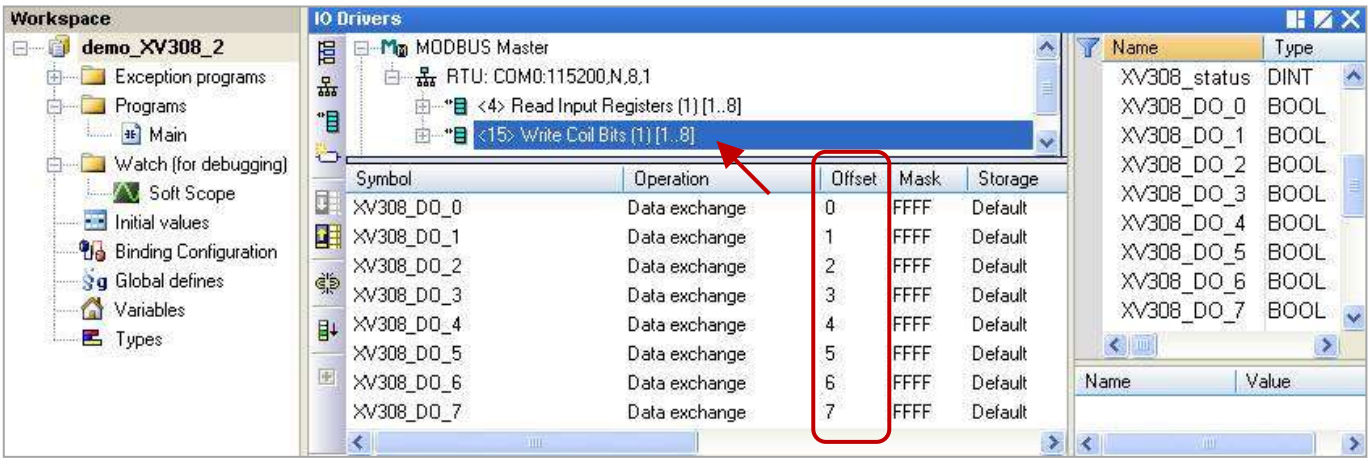
1. Mouse double click the 1st data block (i.e., <4> Read Input Registers) to open the setting window.

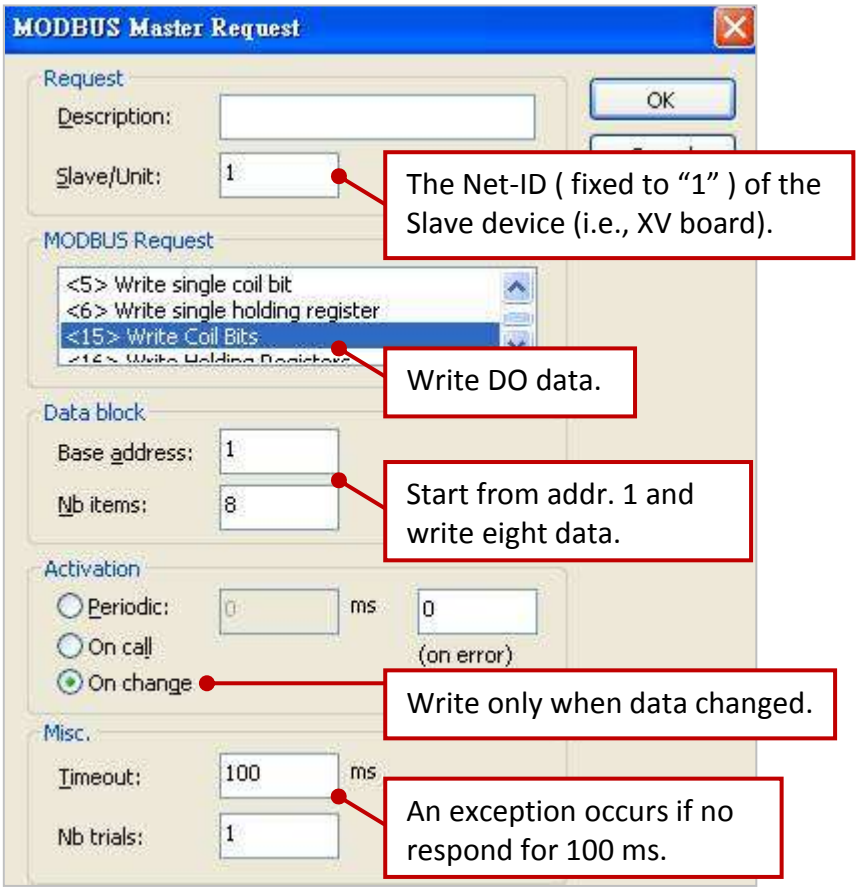


Note: The "Offset" value starts at "0" and the Modbus address of variable is equal to the "Offset" value plus 1 (Base address). Moreover, if you set the "Operation" as "Error report", the "Offset" value for the mapping variable (Date Type: DINT) must set to "0".



2. Mouse double click the 2nd data block (i.e., <15> Write Coil Bits) to open the setting window.

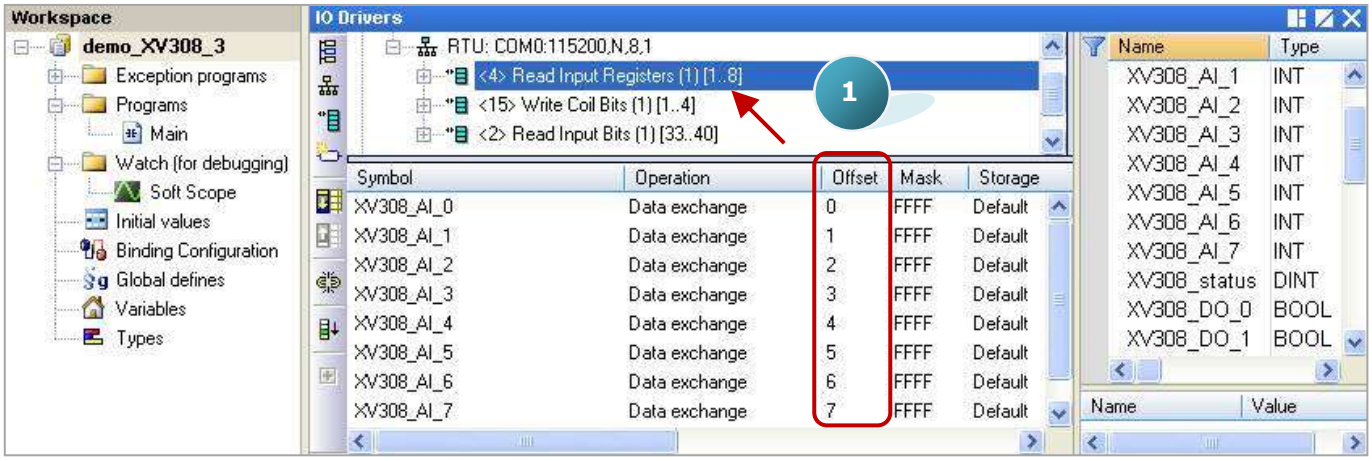




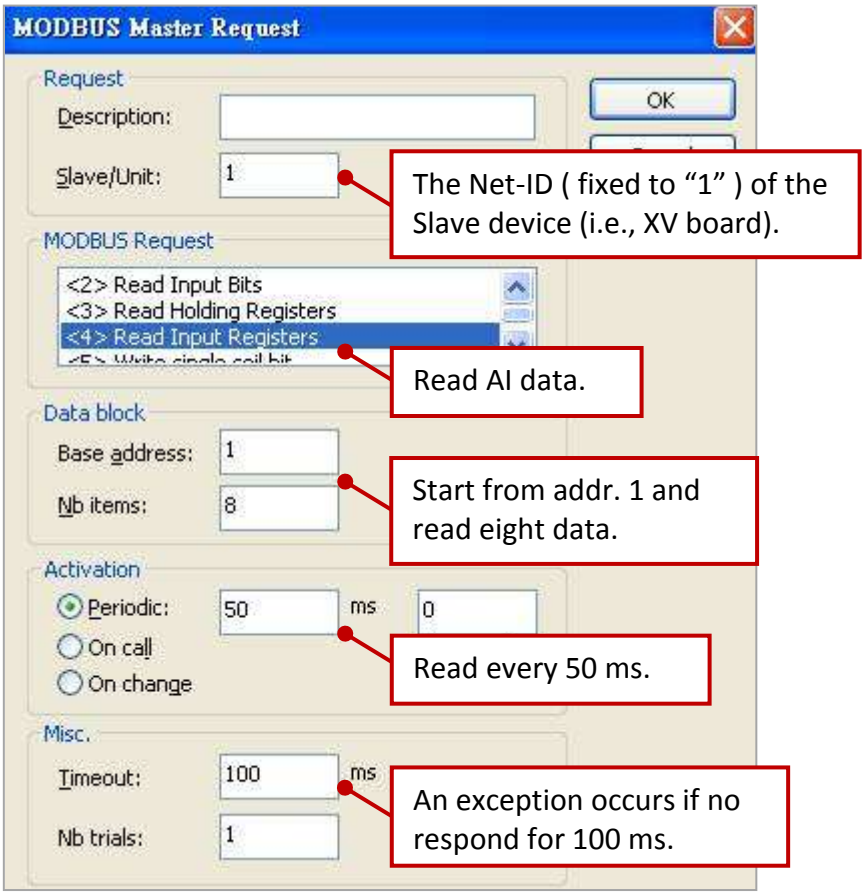
Demo description: (demo_XV308_3)

This demo added three data blocks, the 1st one is used to read 8 AI data, the 2nd one is used to write 4 DO data and the 3rd one is used to read only 4 DI data.

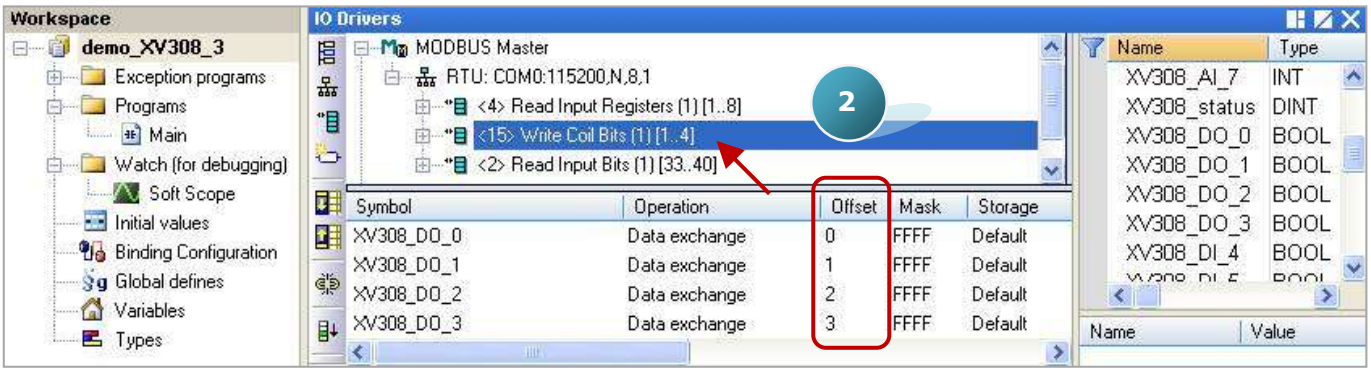
1. Mouse double click the 1st data block (i.e., <4> Read Input Registers) to open the setting window.

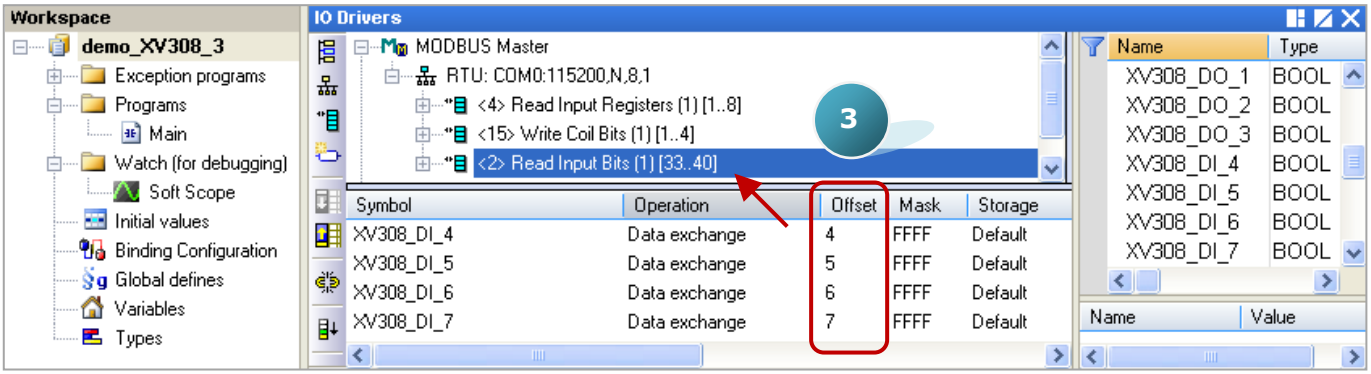


Note: The "Offset" value starts at "0" and the Modbus address of variable is equal to the "Offset" value plus 1 (Base address).

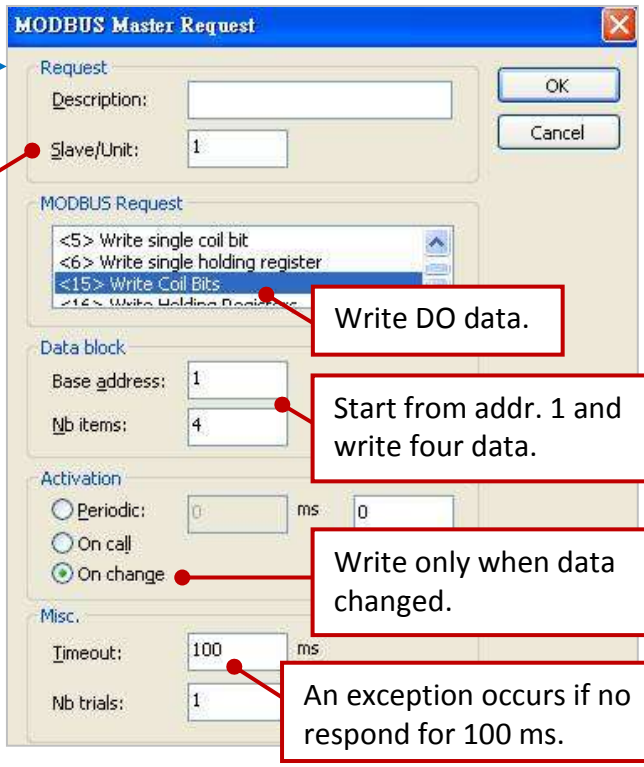


2. As the figure below, mouse double click the 2nd data block (i.e., <15> Write Coil Bits) to view the setting window.
3. As the figure below, mouse double click the 3rd data block (i.e., <2> Read Input Bits) to view the setting window.





2. <15> Write Coil Bits: Write 4 DO data.



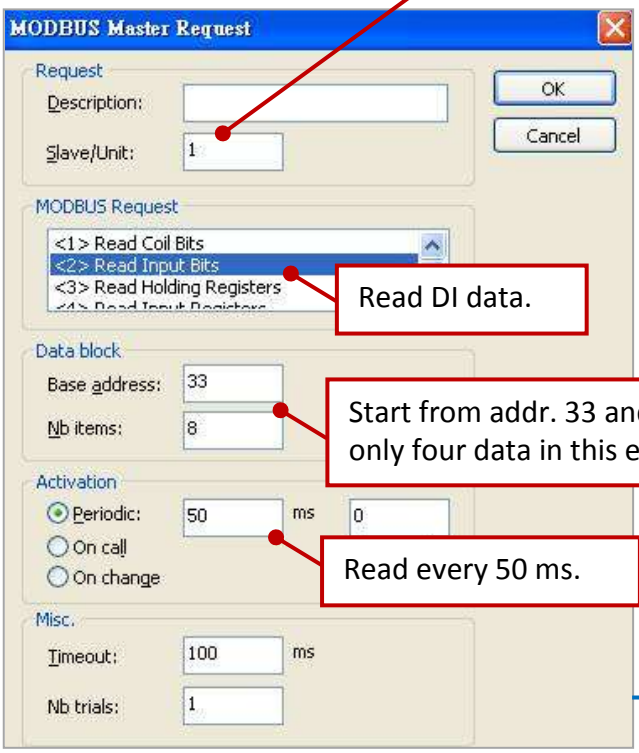
The Net-ID (fixed to "1") of the Slave device (i.e., XV board).

Write DO data.

Start from addr. 1 and write four data.

Write only when data changed.

An exception occurs if no respond for 100 ms.



Read DI data.

Start from addr. 33 and read only four data in this example.

Read every 50 ms.

3. <2> Read Input Bits, read 4 DI data.

Note:

When using the XV308 to read DI data, the address must start from "33".

5.1.12 Connecting the XV310 (4 AI, 2 AO, 4 DI, 4 DO)

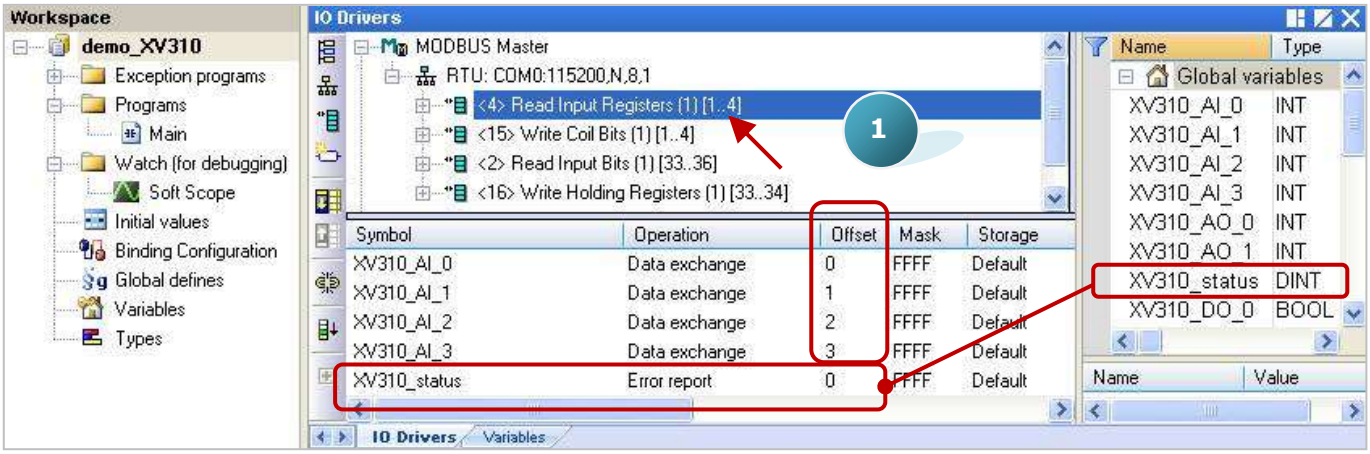
The XV310 is a 4-ch analog input, 2-ch analog output, 4-ch digital input and 4-ch digital ouput board. This section provides a Win-GRAF demo projects - "demo_XV310.zip".

First, go to Section 5.1.6 to view the XV Board instructions and then configure each AI channel by using "DCON_Utility_Pro_CE_200.exe".

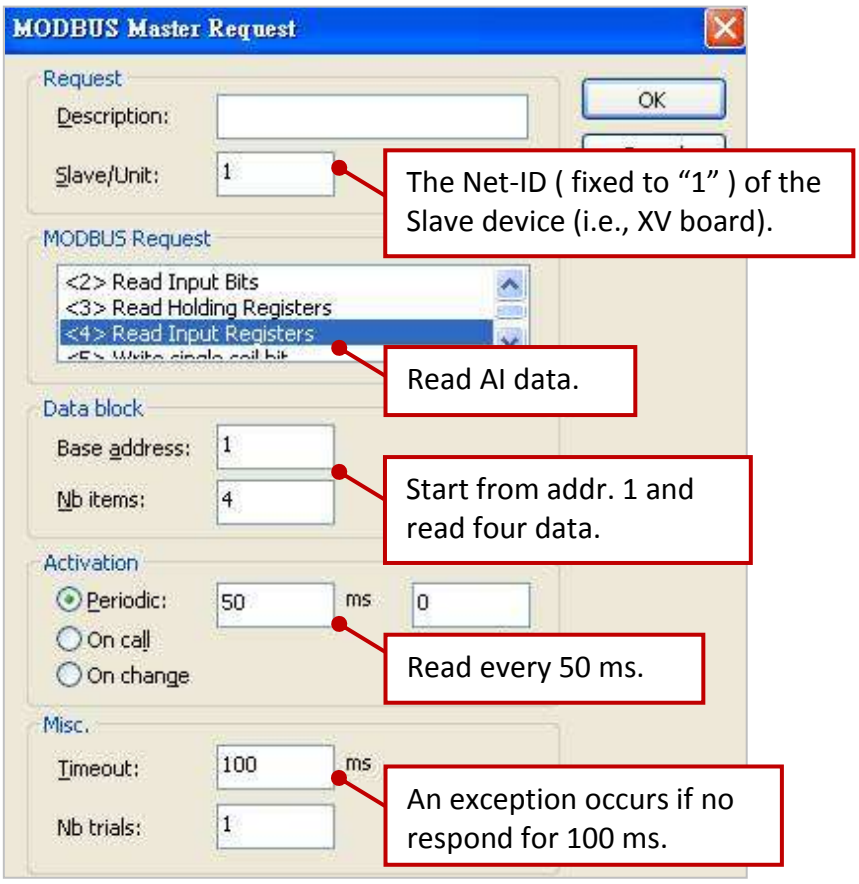
Demo Description

This demo added four data blocks. The 1st one is used to read 4 AI data, the 2nd is used to write 4 DO data, the 3rd is used to read 4 DI data and the 4th is used to write 2 AO data.

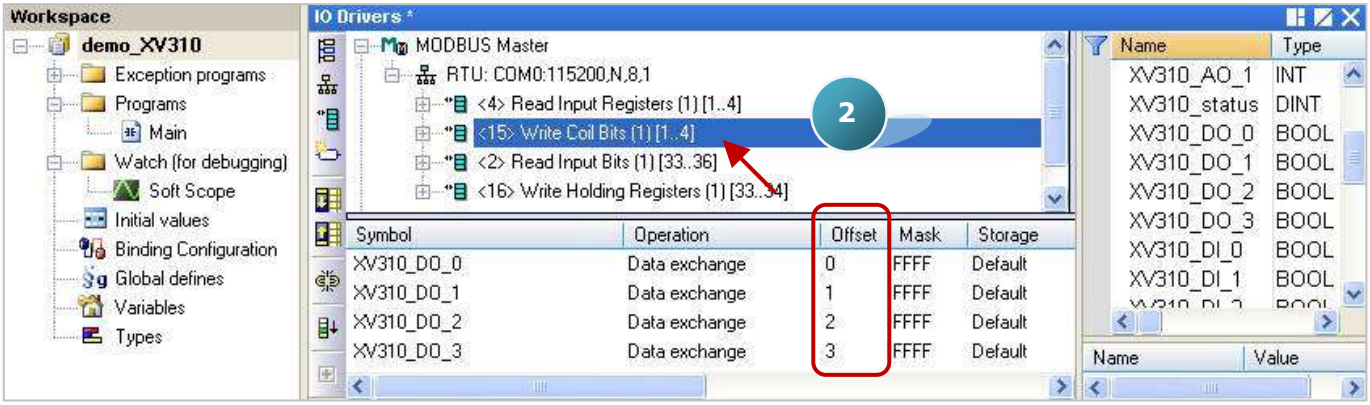
1. Mouse double click the 1st data block (i.e., <4> Read Input Registers) to open the setting window.

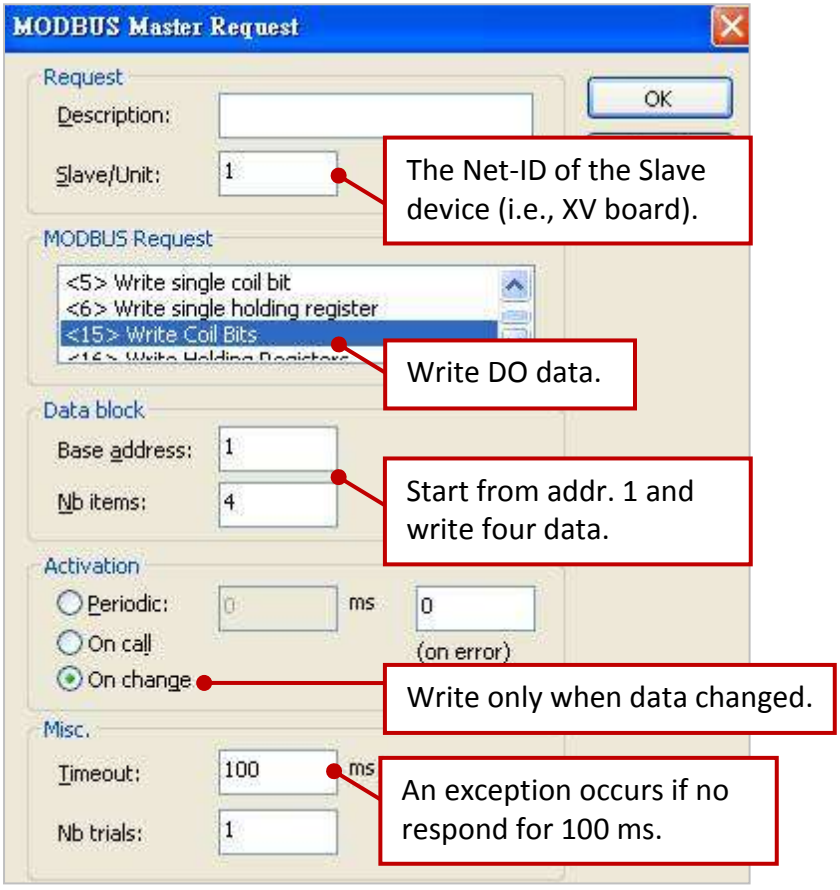


Note: The "Offset" value starts at "0" and the Modbus address of variable is equal to the "Offset" value plus 1 (Base address). Moreover, if you set the "Operation" as "Error report", the "Offset" value for the mapping variable (Date Type: DINT) must be set to "0".

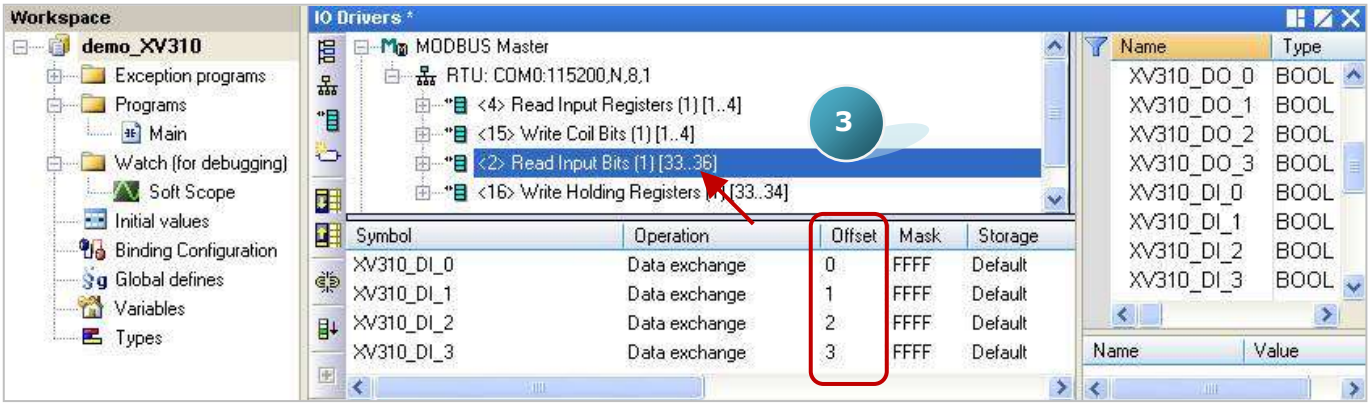


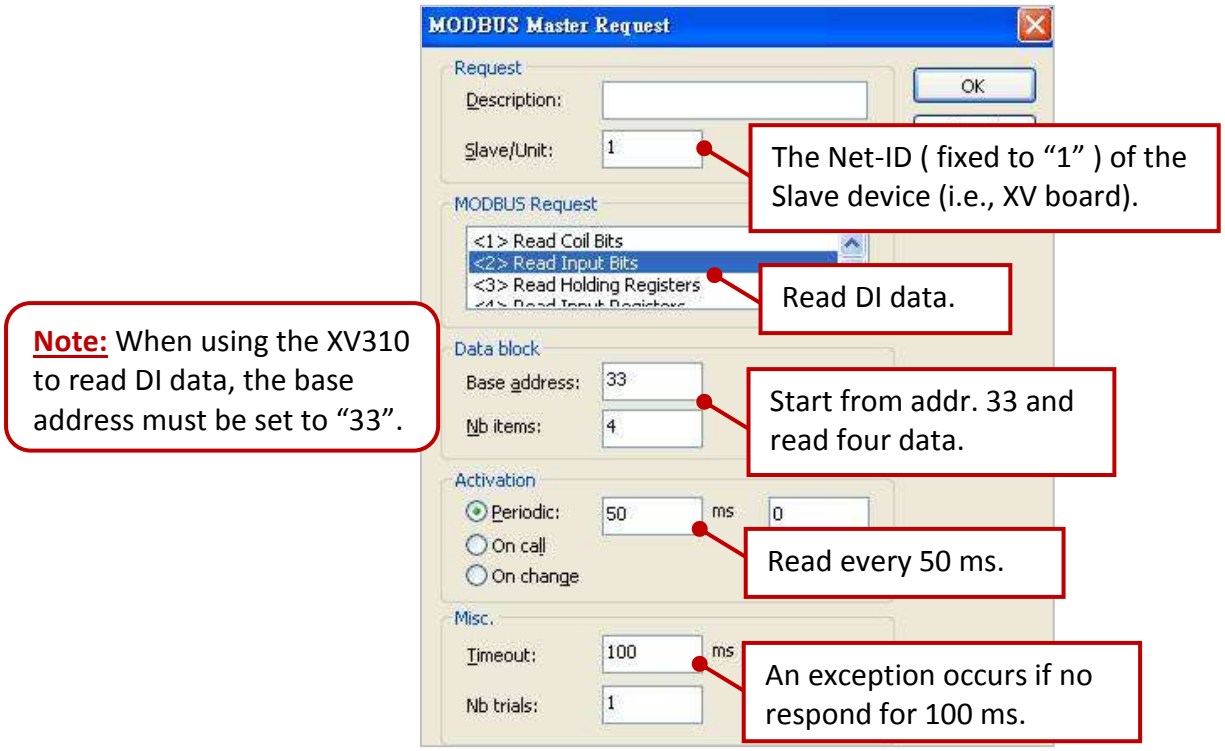
2. Mouse double click the 2nd data block (i.e., <15> Write Coil Bits) to view the setting window.



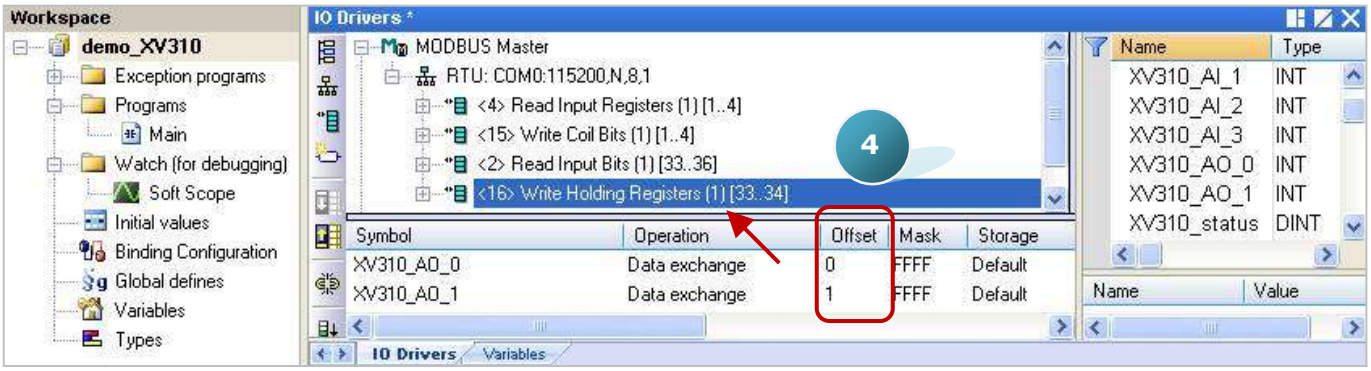


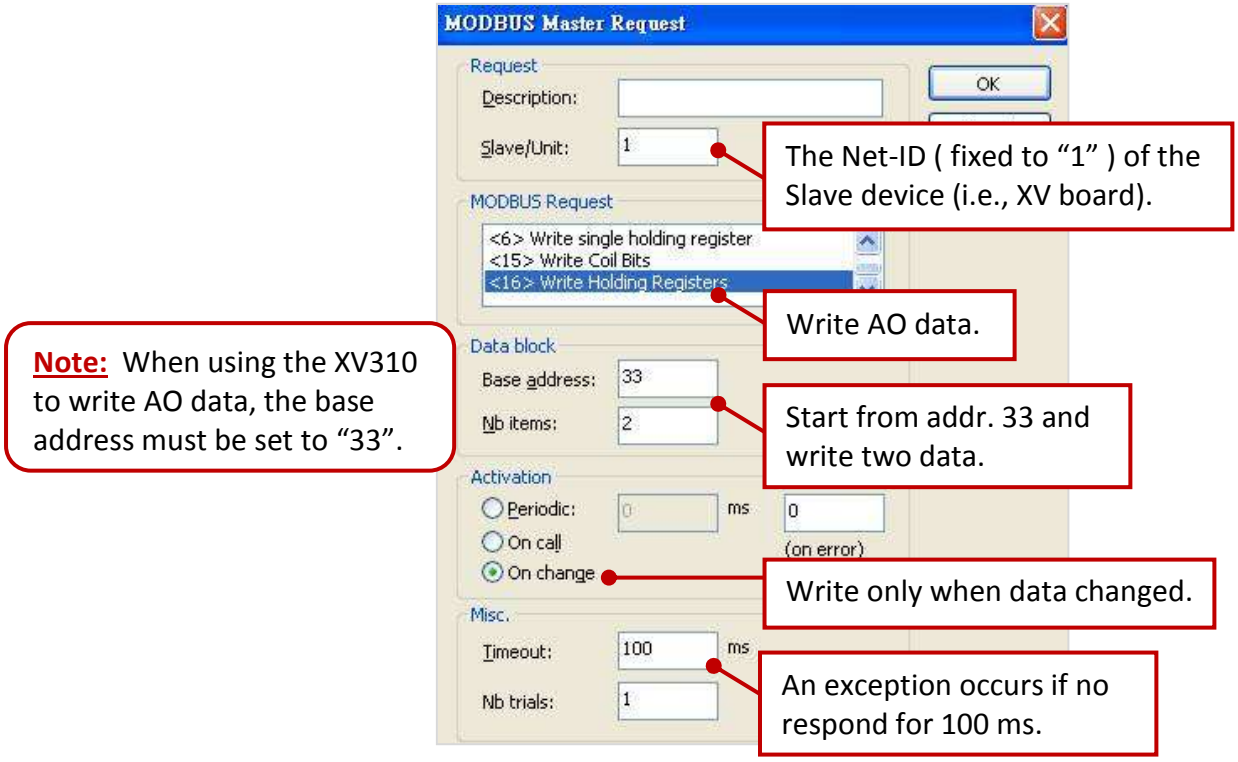
3. Mouse double click the 3rd data block (i.e., <2> Read Input Bits) to view the setting window.





4. Mouse double click the 4th data block (i.e., <16> Write Holding Registers)





5.1.13 To Disable/Enable the Modbus RTU/ASCII Master Port

The Modbus RTU/ASCII Master ports which are enabled in the Win-GRAF "Fieldbus Configuration" - "IO Drivers" setting window, will automatically work after the PAC is powered on. If user wants to disable one of the Modbus Master ports, use the **"MBRTU_M_disable"** function (see below).

```
(* Declare To_disable as BOOL *)
If  To_disable  then
  To_disable := FALSE ;
  MBRTU_M_disable (3) ;
End_if;
```

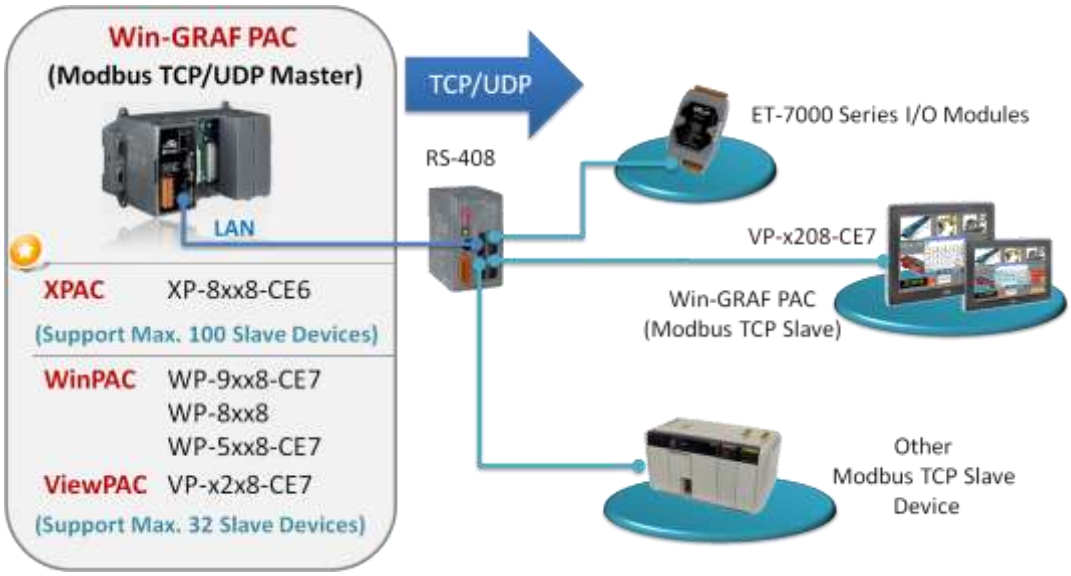
In the above code, when you set "To_disable" as "TRUE", it will disable the Modbus RTU/ASCII Master port - COM3. And later, you can enable it again by using the **"MBRTU_M_enable"** function (see below).

```
(* Declare  To_enable as BOOL
   Declare Status_com3 as BOOL *)
If  To_enable  then
  To_enable := FALSE ;
  MBRTU_M_enable (3) ;
End_if;
Status_com3 := MBRTU_M_status (3) ;
```

The **"MBRTU_M_status"** function listed above is used to get the status of the Modbus RTU/ASCII Master port, for example, enabled (True) or disabled (False).

5.2 Enabling the Win-GRAF PAC as a Modbus TCP/UDP Master (Ethernet I/O)

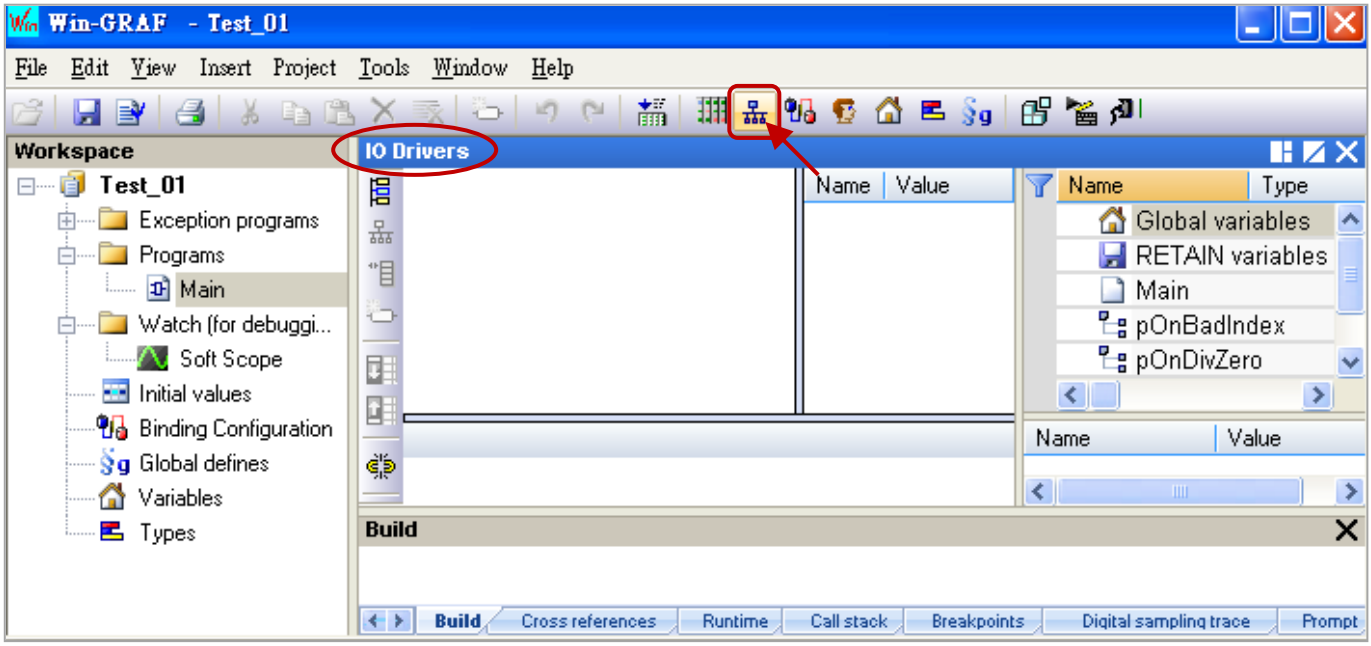
Application Diagram:



(Refer P1-1 to see the PAC model numbers.)

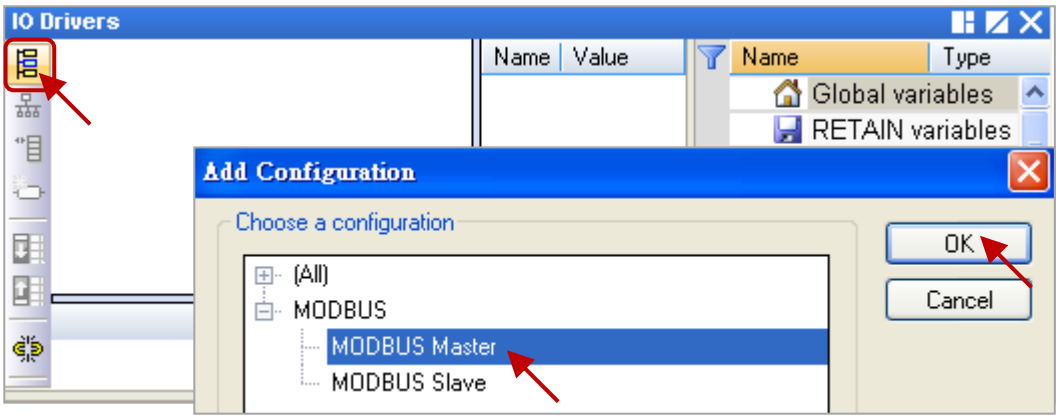
The Setting Steps:

1. Click the tool icon "Open Fieldbus Configuration" to open the "I/O Drivers" window.



2. Click "Insert Configuration" icon in the left side of the "I/O Drivers" window, and then click "MODBUS Master", then click "OK" to enable a Modbus Master.

Note: One "Modbus Master" can set up multiple Ports (see the next step), can set as a Modbus Master RTU/ASCII Port (Refer Section 5.1) or a Modbus Master TCP/UDP Port or can set up not to enable the setting.



3. Click the tool icon "Insert Master/Port" in the left side, open the setting window and select the "MODBUS on Ethernet".

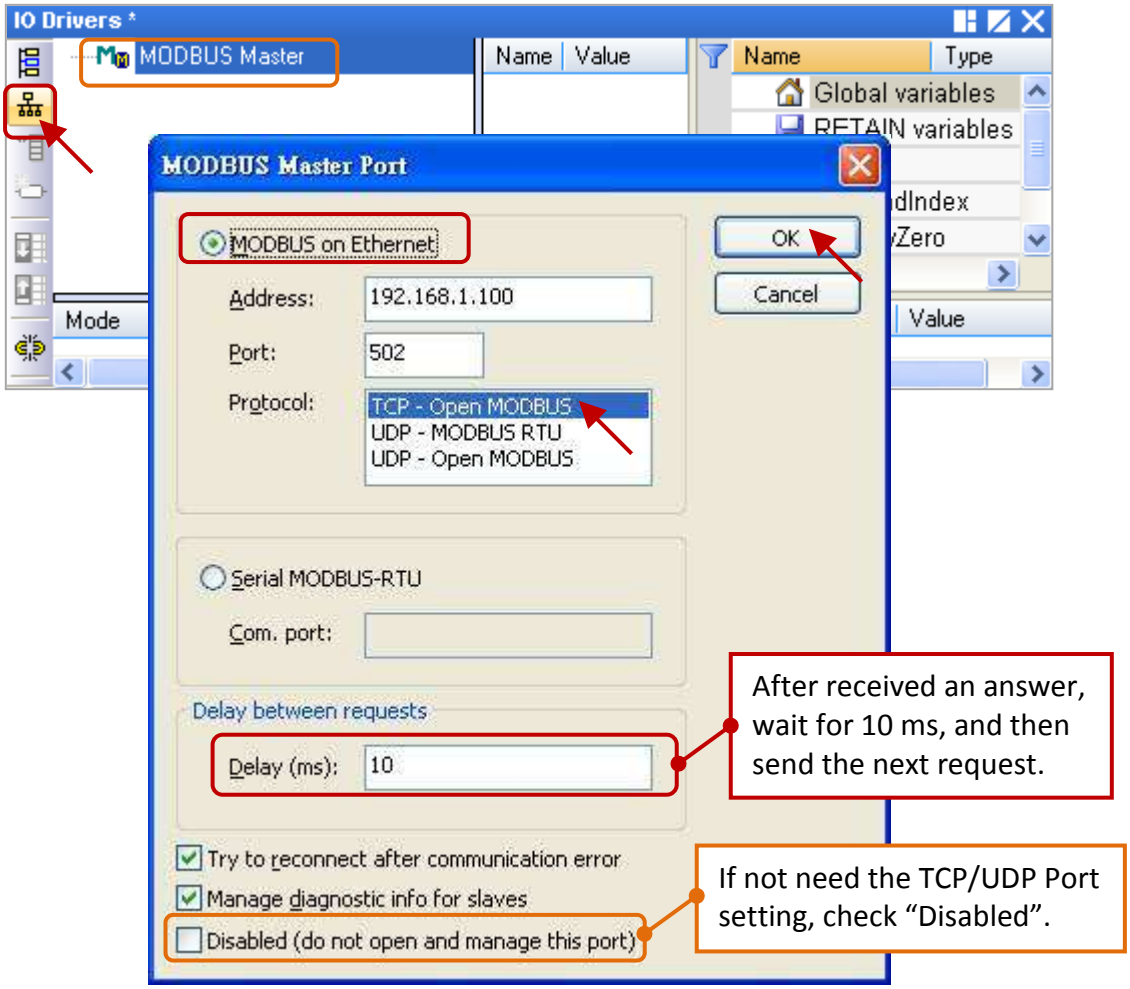
   Set up the following items, and then click "OK".

   Address: Fill in the IP Address of the Modbus Slave device (e.g., "192.168.1.100").

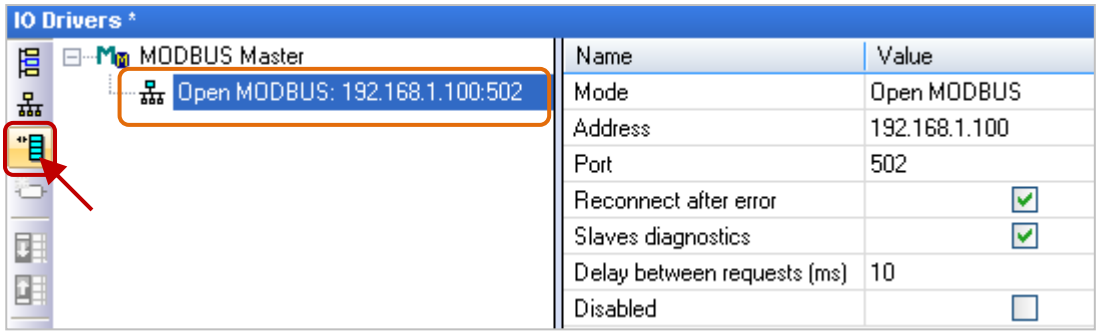
   Port: TCP port Number of the Slave device.

   Protocol: If as a Modbus TCP Master, select the "TCP – Open MODBUS".
   If as a Modbus UDP Master, choose the "UDP – Open MODBUS".

   Delay: Fill in the delay time (e.g., 10 ms, can be 0 ~ 10000).

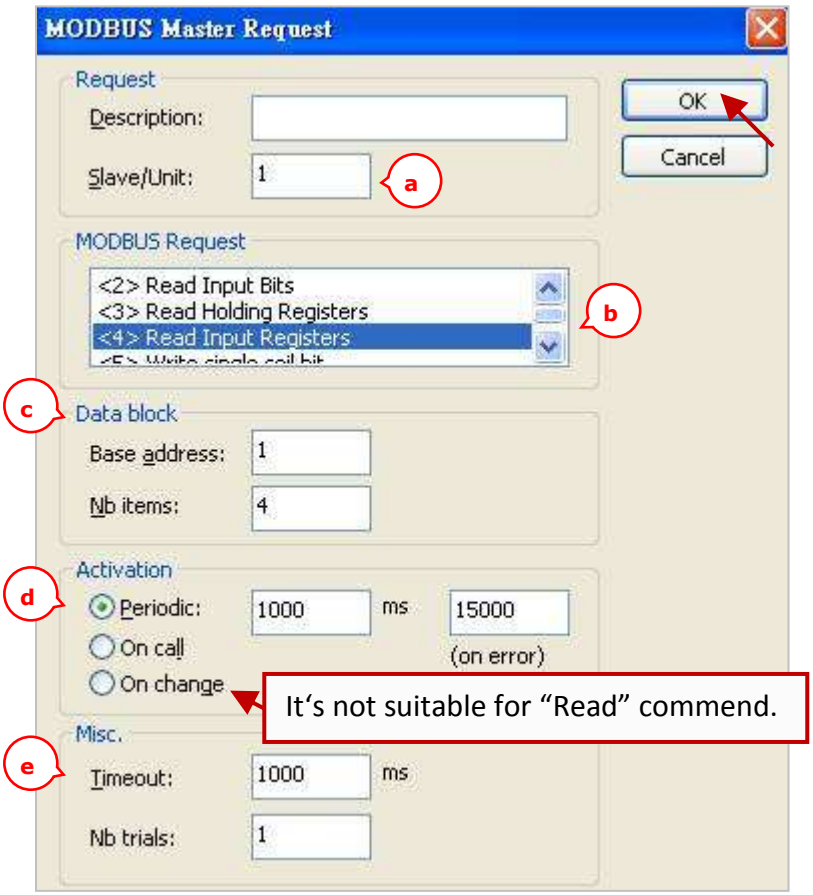


4. Click the icon “Insert Slave/Data Block” in the left side to create a “Data Block”.



Read AI Data

5. In the “MODBUS Master Request” setting window, set up the following items, and then click “OK".



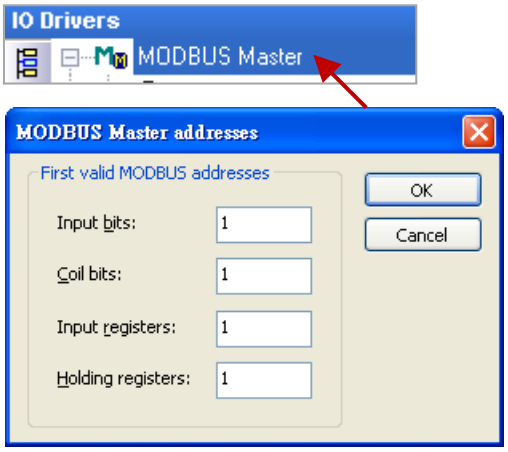
In this example

a. Slave/Unit:
   Fill in the Net-ID of the Slave device (Usually is “1”).

b. MODBUS Request:
   Select “<4> Read Input Registers”.

c. Base address:
   Default to start from 1.
   Nb items:
   The AI numbers to read (here is 4).

Note:
If want to change the “Base address”, please use mouse to right-click the “MODBUS Master”, and then select “MODBUS Master Addresses” to change the value.



d. Activation: the sending way of Modbus Request.

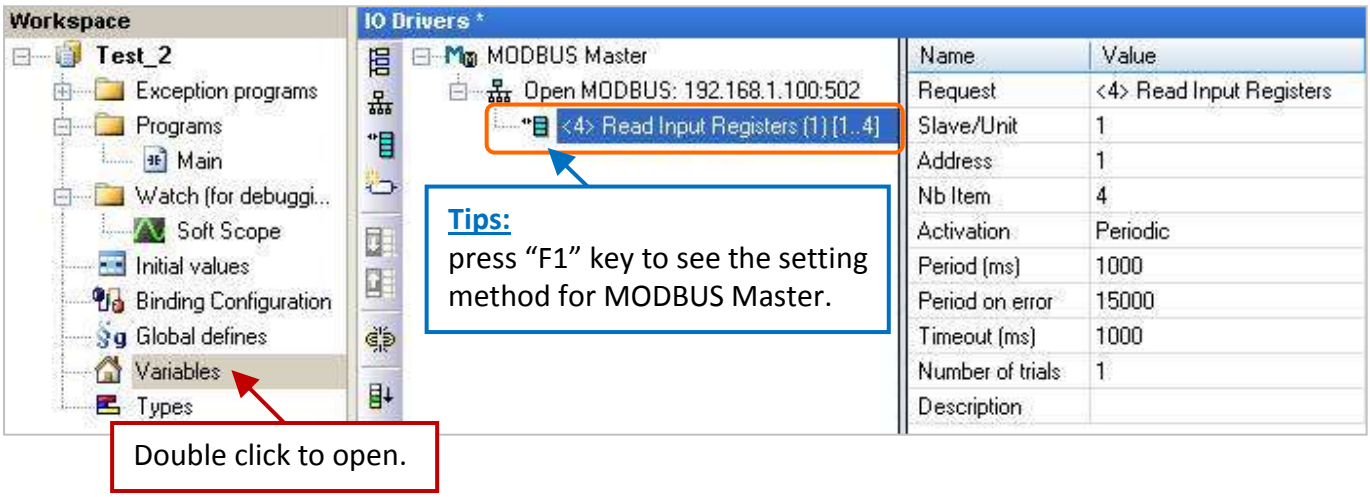
Periodic: Send request periodically. In this case, it sends request every 1 Sec. “on error” means that when an error occurs, the next sending time (in this case, 15 seconds).

On call: It will send the request once when a program calls it.

On change: It will send the request once when data is changed.

e. Timeout: Set up the max. time to wait for the response. If exceeds it, that means an error. (For Modbus TCP/UDP, recommended: 1000 ~ 3000 ms; this example is 1000 ms)

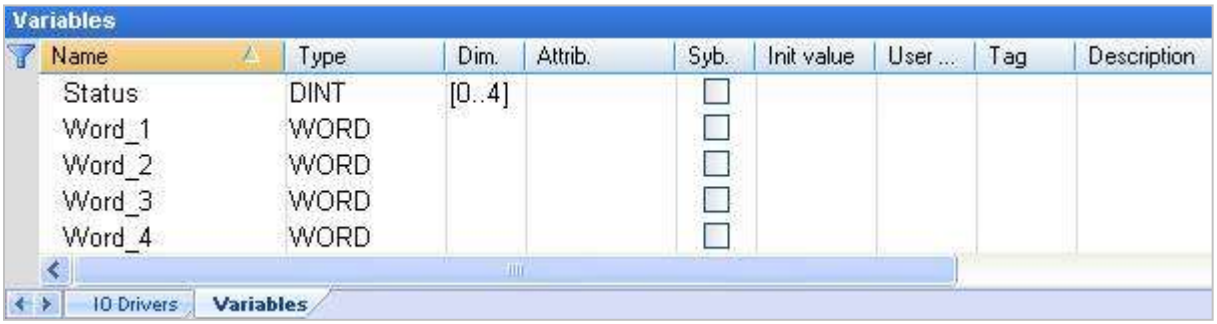
6. Open the “Variables” window, set up the variables want to use.



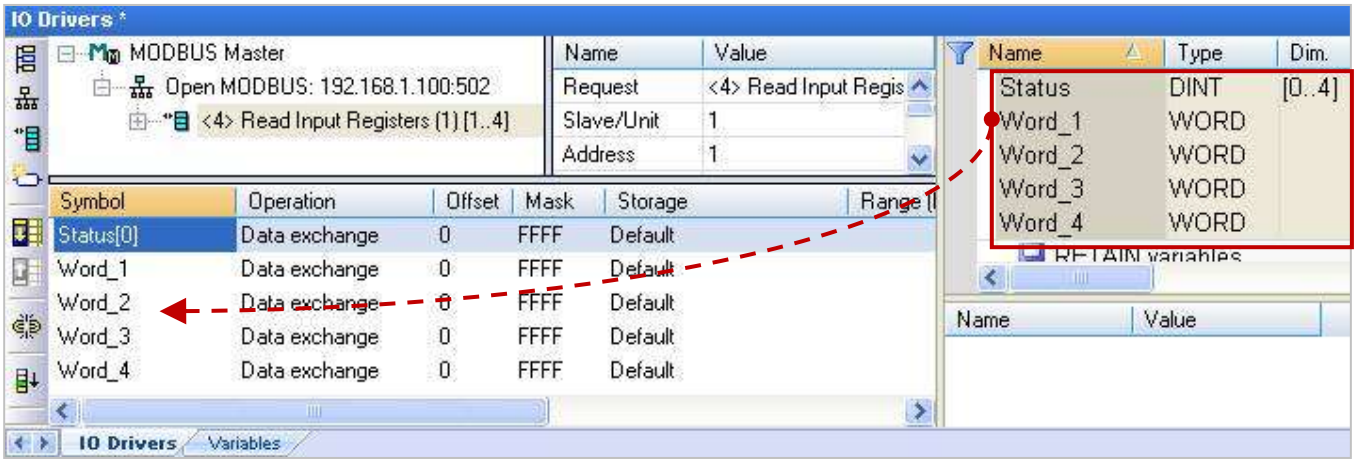
Please follow the table to set up 4 WORD (16 bit) variables (refer Section 2.3.1).

Variable Name	Data Type	Dim.	Description
Word_1 ~ Word_4	WORD	---	Used to read the AI data (16 bit)
Status	DINT	5	Used to record the read/write status

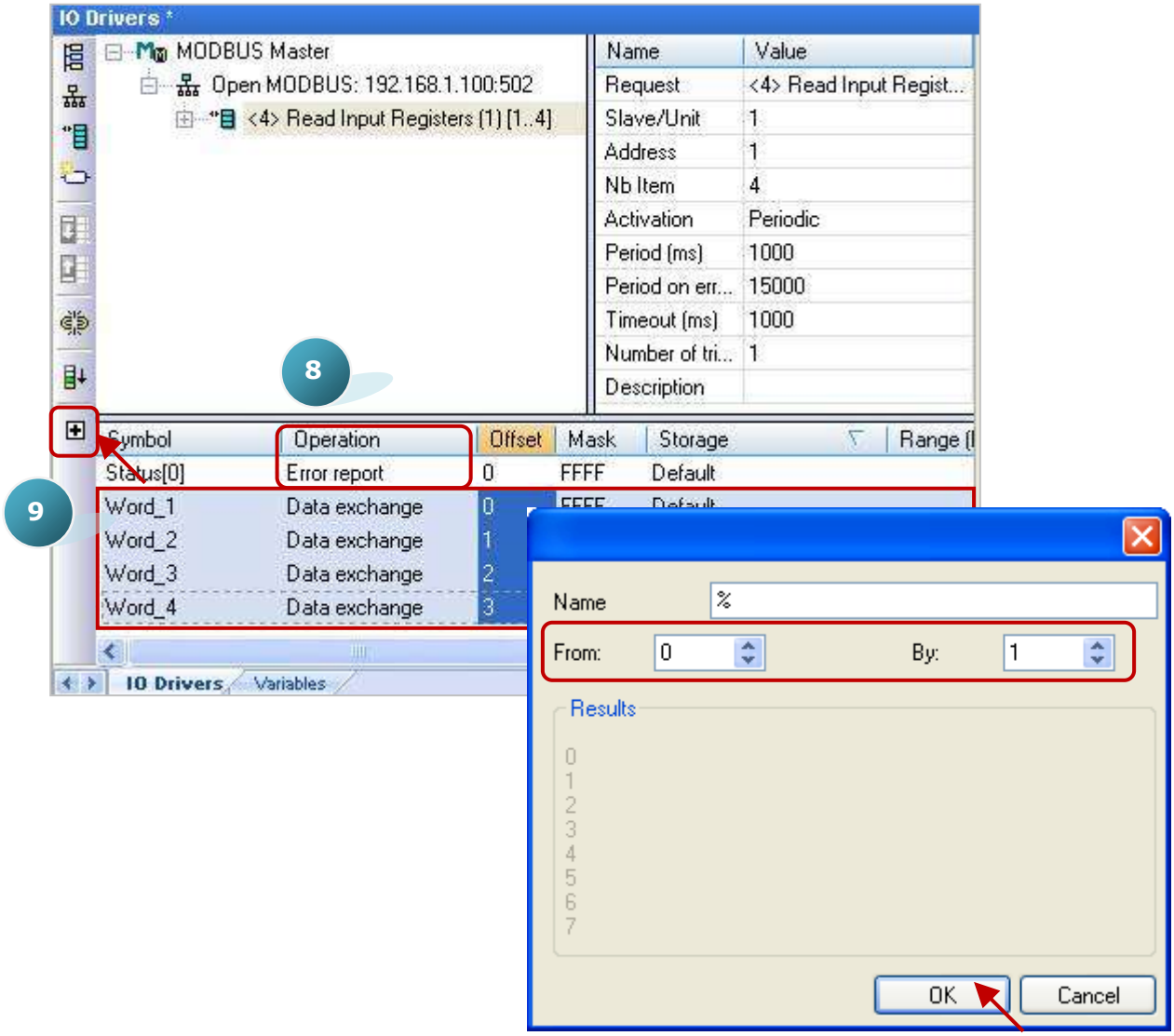
Refer the Appendix A to see the data types and range of the variables. After setting up, it is as the picture below.



7. In the “I/O Drivers” window, drag the variables (“Word_1 ~ Word_4” and “Status”) from the Variables area to the “Symbol” area of the Data Block. **Notice:** This example shows “Status” is an Array variable. When drag it to the “Symbol” area, it will become “Status[0] ~ Status[4]", please press “Delete” key to delete “Status[1] ~ [4]".



8. Set the “Operation” of the “Status[0]” to “Error report” (If reading data fails, its value is an “Error Code”; when reading data OK, it will reset to “0”.). Press “F1” key to view the setting descriptions for the Modbus Master. In the title of “Status and command variables”, you can find the details about this commend and “Error Code”.
9. Select "Word_1 ~ Word_4” and click “Iterate property” to set up the “Offset” value (From: 0; By: 1).



The setting steps of “Modbus Master Request” for both “Modbus Master RTU/ASCII Port (Section 5.1)” and “Modbus Master TCP/UDP Port” are the same. Now, we have finished the setting to read AI data. Please click the item number (link to the Section 5.1.1~5.1.5) in the table below for the setting steps to read/write other data.

Items	Function Code	Modbus Request	Description
1	2	Read Input Bits	Read DI data
2	5	Write single coil bit	Write DO data
3	4	Read Input Registers	Read AI data
4	6	Write single holding register	Write one AO data (16-bit)
5	16	Write Holding Registers	Write multiple AO data (16/32 bits)

Note: If you want to disable the Modbus TCP/UDP Master port while the program is running. Refer the Section 5.2.4 to use the "MBTCP_M_disable" function (and use “MBUDP_M_disable” for UDP).

5.2.1 Connecting ET-7000 Series I/O Module

ICP DAS ET-7000 is a series of I/O module supporting Modbus TCP Slave protocol. The Win-GRAF PAC can enable the Modbus TCP Master to connect the ET-7000 modules. The maximum recommend the amount of the connecting ET-7000 modules depends on the PAC model, such as the WP-5238-CE7 and XP-8xx8-CE6, recommends a maximum of 200; the WP-8xx8, WP-8xx8-CE7, and VP-x2x8-CE7 is recommended that no more than 32.

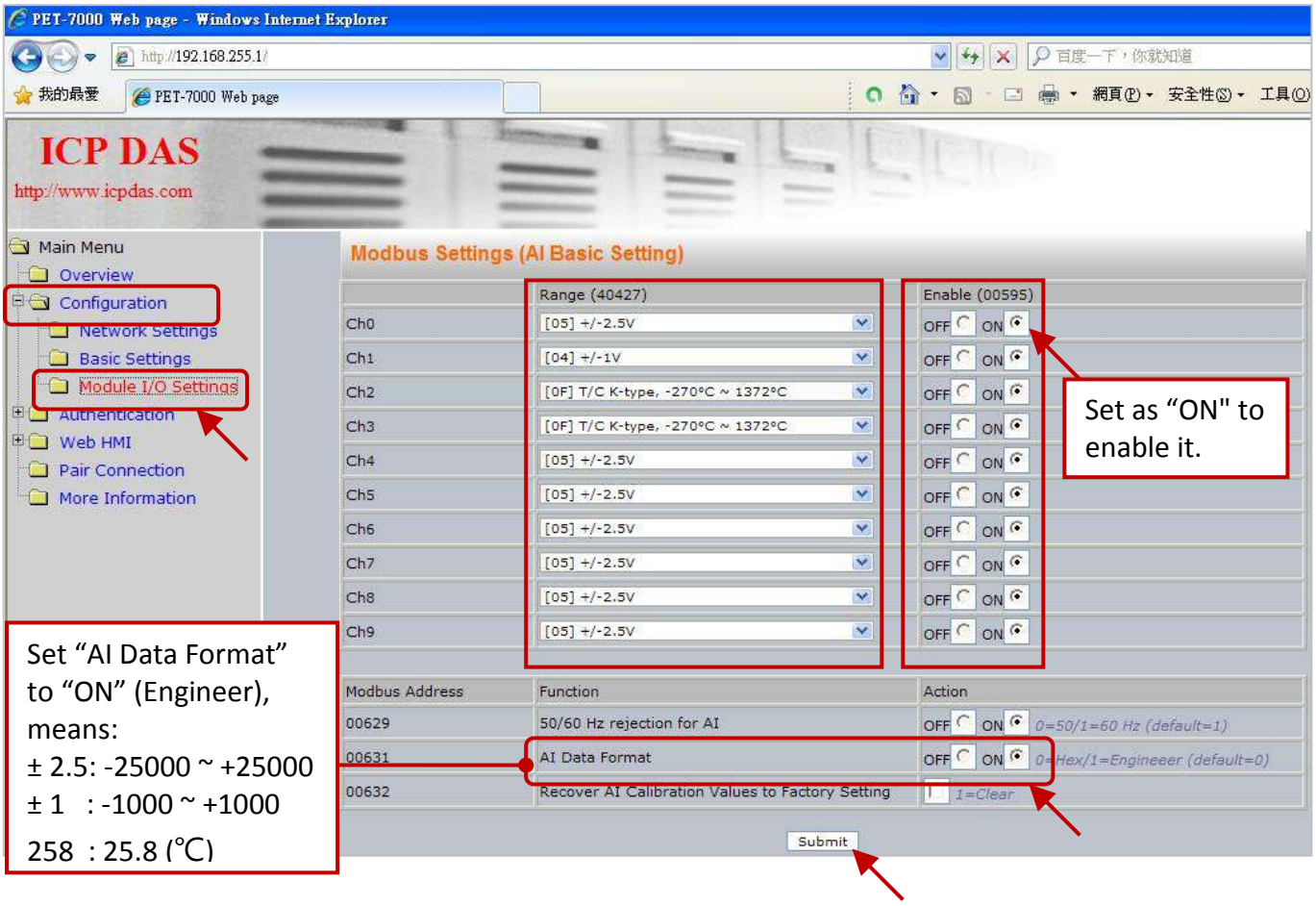
For more information about the ET-7000 series products, please visit the website:
http://www.icpdas.com/root/product/solutions/remote_io/ethernet_io/ethernet_io_selection.html

Use Internet Browser to Set the ET-7000 Modules

Before the first time using the ET-7000, you must set up the ET-7000 by using the Internet Browser. When the ET-7000 shipping from the factory, the settings are: IP address = 192.168.255.1; Mask = 255.255.0.0. Please set the IP of your PC in the same network (e.g., set the IP to 192.168.255.100, Mask = 255.255.0.0), then open the browser (such as IE), and enter the IP of the ET-7000 to connect it.

Notice: The Dip Switch on the rear of the ET-7000 must stay in the “Normal” position.

Click “Configuration” > “Module I/O Settings” to set up the range of channels as below, and then click “Submit”.



Users can set the ET-7018Z‘s "AI Data Format" to "ON" (Engineering) for more convenient usage.
For example:

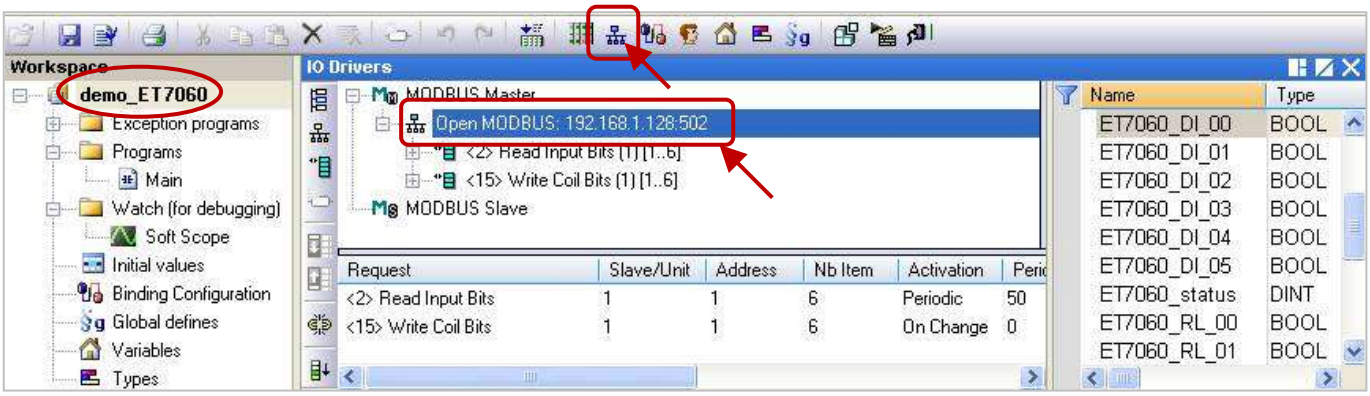
Type Code	Range	Data Format	Minimum	Maximum
04	-1 ~ +1 V	Engineering	-10000	+10000
		2’s comp HEX	8000h	7FFFh
05	-2.5 ~ +2.5 V	Engineering	-25000	+25000
		2’s comp HEX	8000h	7FFFh
18	Type M Thermocouple -200 ~ 100°C	Engineering	-20000	+10000
		2’s comp HEX	8000h	4000h

Restore/Open the Demo Project:

The Win-GRAF demo projects in the following sections can be found on the shipping CD, please refer Chapter 12. Click the menu bar "File" > "Add Existing Project" > "From Zip" can restore/open/check the demo projects. (CD-ROM:\Napdos\Win-GRAF\demo-project\)

Demo Project	File Name	Description
ET-7060	demo_ET7060.zip	Read 6 DIs, write 6 DOs
ET-7018Z	demo_ET7018z.zip	Read 10 AIs

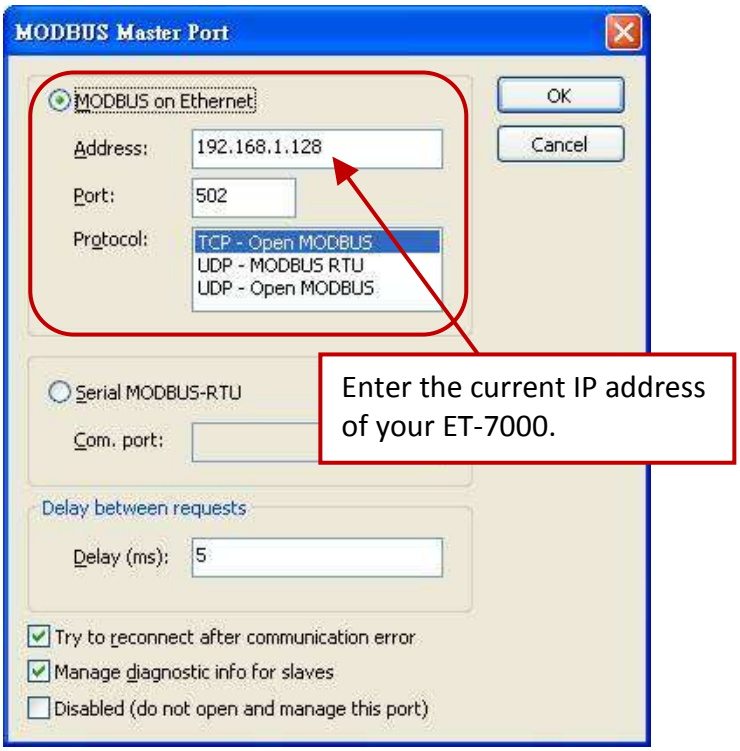
1. Click the tool icon “Open Fieldbus Configuration” to open the “I/O Drivers” window.



2. Double click “Open Modbus: IP:502” to open the “MODBUS Master Port” window.

Notice:

All demo projects in this chapter can enable the Win-GRAF PAC as a Modbus **TCP** Master. Please fill in the current IP address of your ET-7000, and set "Port" to "502” and "Protocol" to "TCP - Open Modbus".



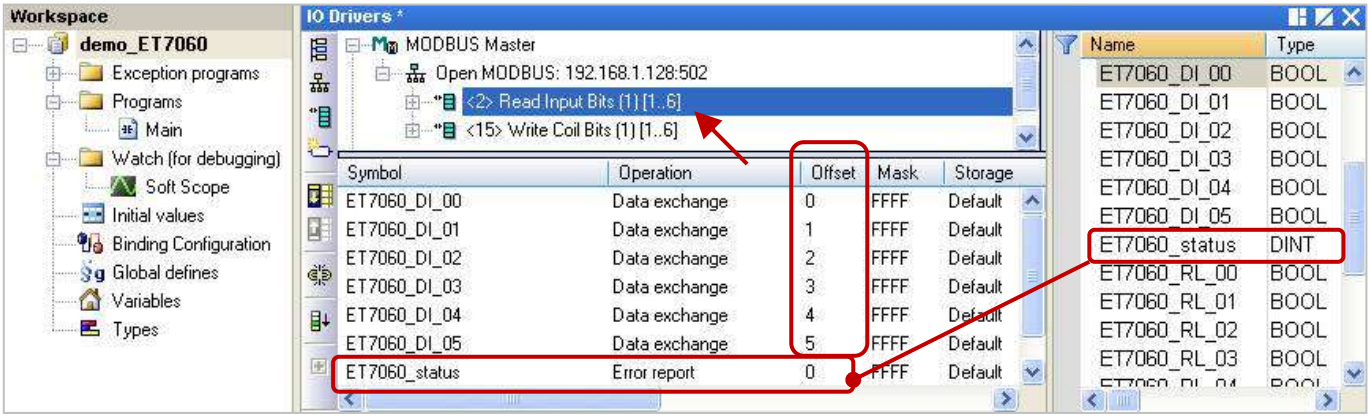
5.2.2 Connecting the ET-7060 (6 DI, 6 Relay)

The ET-7060 is a 6 DI and 6 Relay channels Ethernet I/O module. The Win-GRAF demo project for this section is "demo_ET7060.zip". Please refer Section 5.2.1 to set up the module channels using the Internet Browser, and restore/open the demo project.

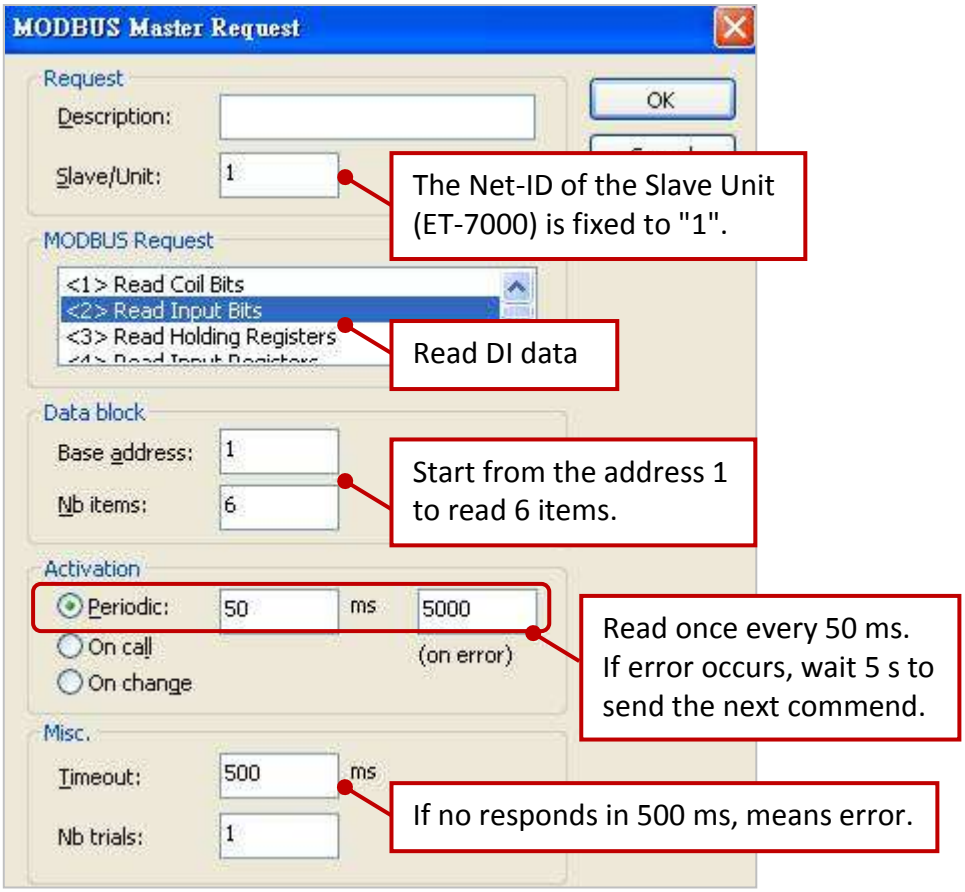
Demo Description:

This demo creates two Data Blocks, one is used to read 6 DI data, the other is used to write 6 DO data.

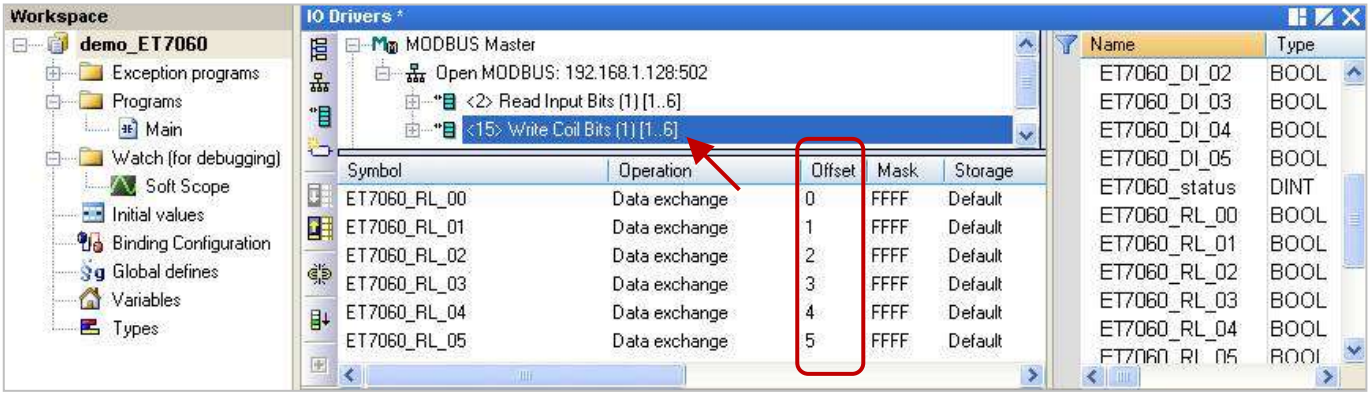
1. Double click the first Data Block (<2> Read Input Bits) to open the setting.

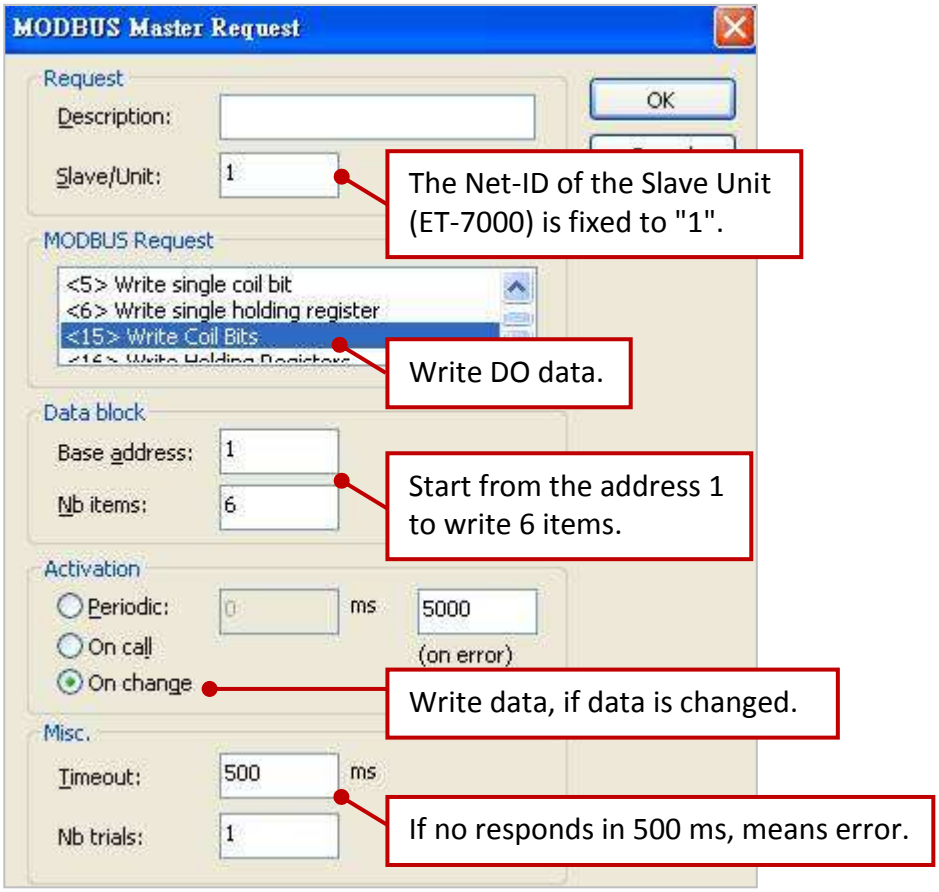


Notice: The value of the "Offset" starts from "0", but the Modbus address of the variable is the "Offset" value plus 1 (Base address). If set the "Operation" to "Error report", the "Offset" value of the variable (Data Type: DINT) must set to "0".



2. Double click the second Data Block (<15> Write Coil Bits) to open the setting window.





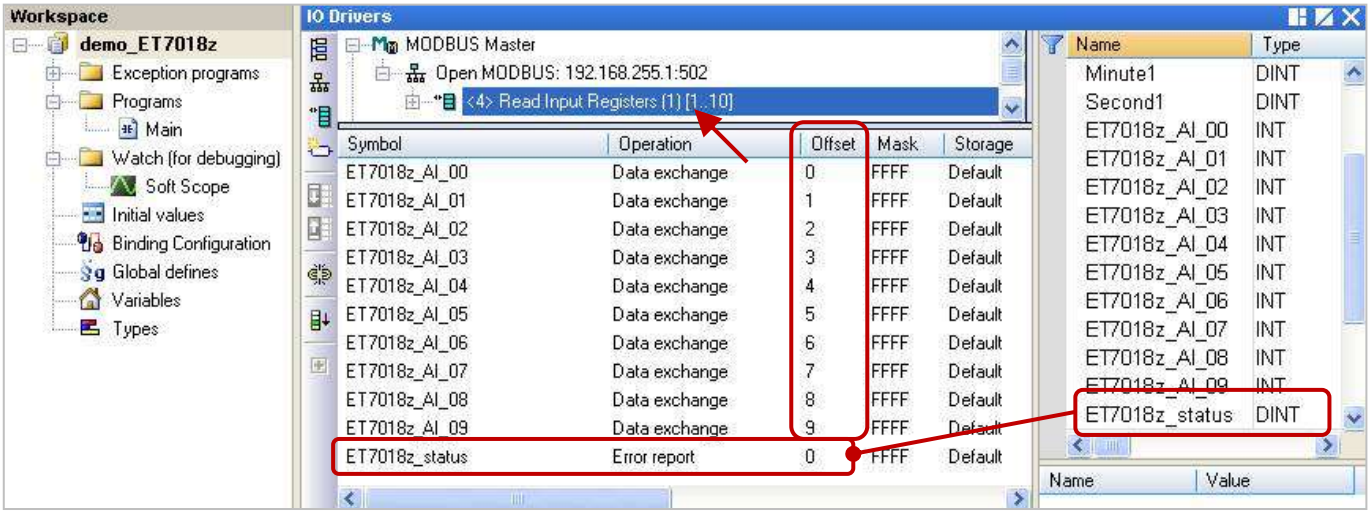
5.2.3 Connecting the ET-7018Z (10 AI)

The ET-7018Z is an 10 AI channels Ethernet I/O module. The Win-GRAF demo project for this section is "demo_ ET7018z.zip". Please refer Section 5.2.1 to set up the module channels using the Internet Browser, and restore/open the demo project.

Demo Description:

This demo creates one Data Block to read 10 AI data.

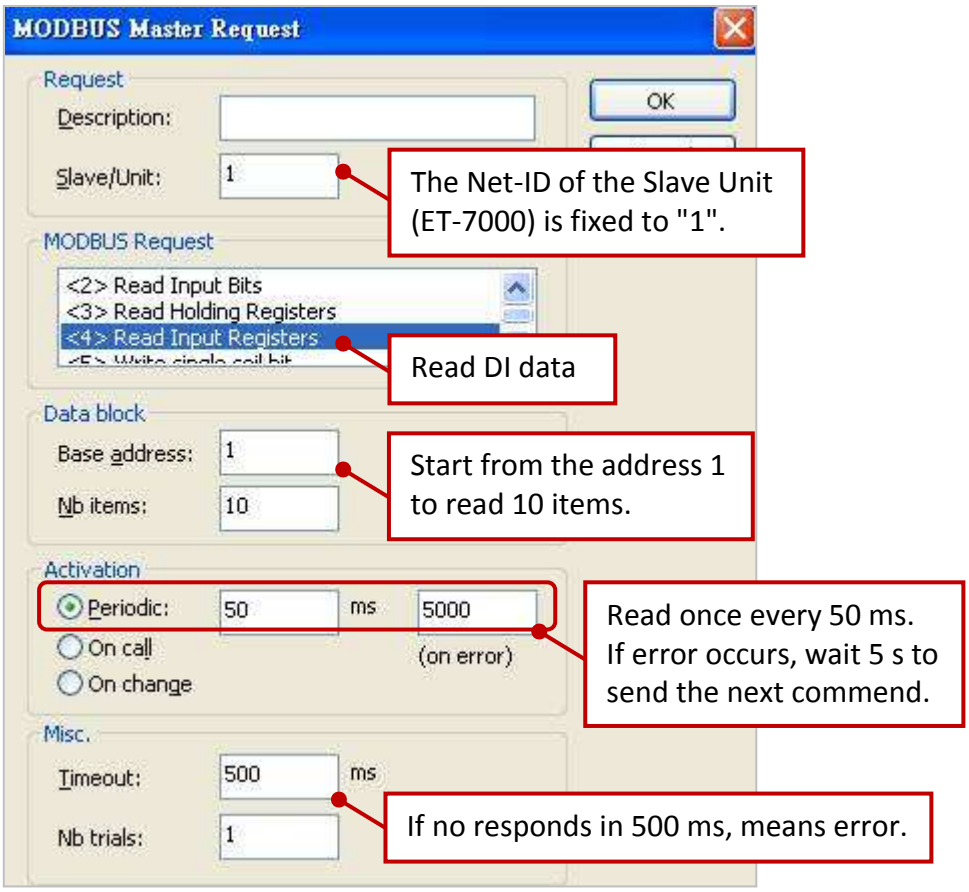
1. Double click the first Data Block (<4> Read Input Registers) to open the setting window.



Notice: 1. The value of the "Offset" starts from "0", but the Modbus address of the variable is the "Offset" value plus 1 (Base address).

2. If set the "Operation" to "Error report", the "Offset" value of the variable (Data Type: DINT) must set to "0".

3. If AI range is -32768 ~ 32767, please declare the data type as "INT" for the variable.



5.2.4 To Disable/Enable the Modbus TCP/UDP Master Port

The Modbus TCP/UDP Master ports which are enabled in the Win-GRAF "Fieldbus Configuration" - "IO Drivers" setting window, will automatically work after the PAC is powered on. If user wants to disable one of the Modbus TCP Master ports, use the **"MBTCP_M_disable"** function (and use the **"MBUDP_M_disable"** function for UDP), see below:

```
(* Declare To_disable as BOOL *)
If  To_disable  then
  To_disable := FALSE ;
  MBTCP_M_disable ( '192.168.71.9' , 502 ) ;
End_if;
```

In the above code, when you set "To_disable" as "TRUE", it will disable the Modbus TCP Master port which connects to the slave device with the IP address "192.168.71.9" (TCP Port_No = 502). And later, you can enable it again by using the **"MBTCP_M_enable"** function (using the **"MBUDP_M_enable"** function for UDP), see below:

```
(* Declare  To_enable as BOOL
    Status_tcp as BOOL *)
If  To_enable  then
  To_enable := FALSE ;
  MBTCP_M_enable ( '192.168.71.9' , 502 ) ;
End_if;
Status_tcp := MBTCP_M_status ( '192.168.71.9' , 502 ) ;
```

The **"MBTCP_M_status"** function (and **"MBUDP_M_status"** is for UDP) listed above is used to get the status of the Modbus TCP Master port, for example, enabled (True) or disabled (False).

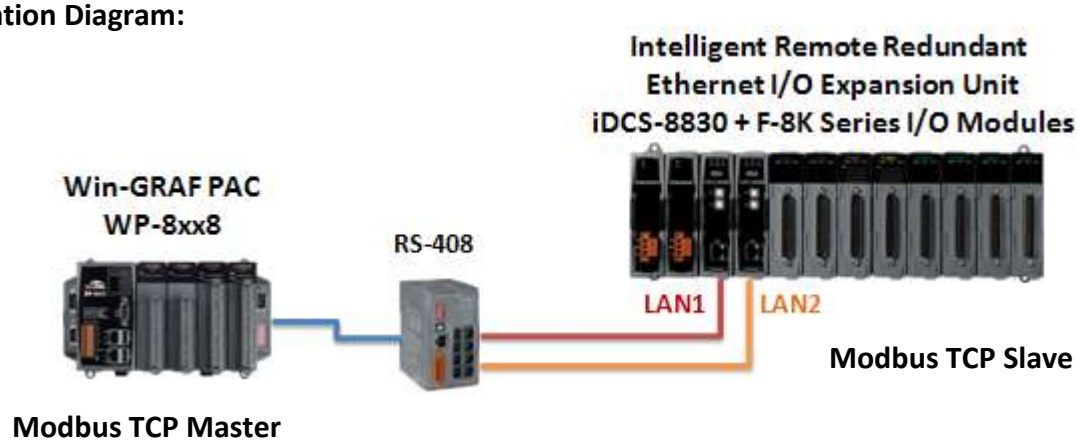
For easy maintenance, user can declare a STRING variable (set its length as "20"). For example, declare one "IP_addr2" variable and set its initial value as "192.168.71.9“. Then you can use it as the following code.

```
If  To_disable  then
   To_disable := FALSE ;
   MBTCP_M_disable ( IP_addr2 , 502 ) ;
End_if;
Status_tcp2 := MBTCP_M_status ( IP_addr2 , 502 ) ;
```

5.3 Connecting the Modbus TCP Slave device has two IP addresses

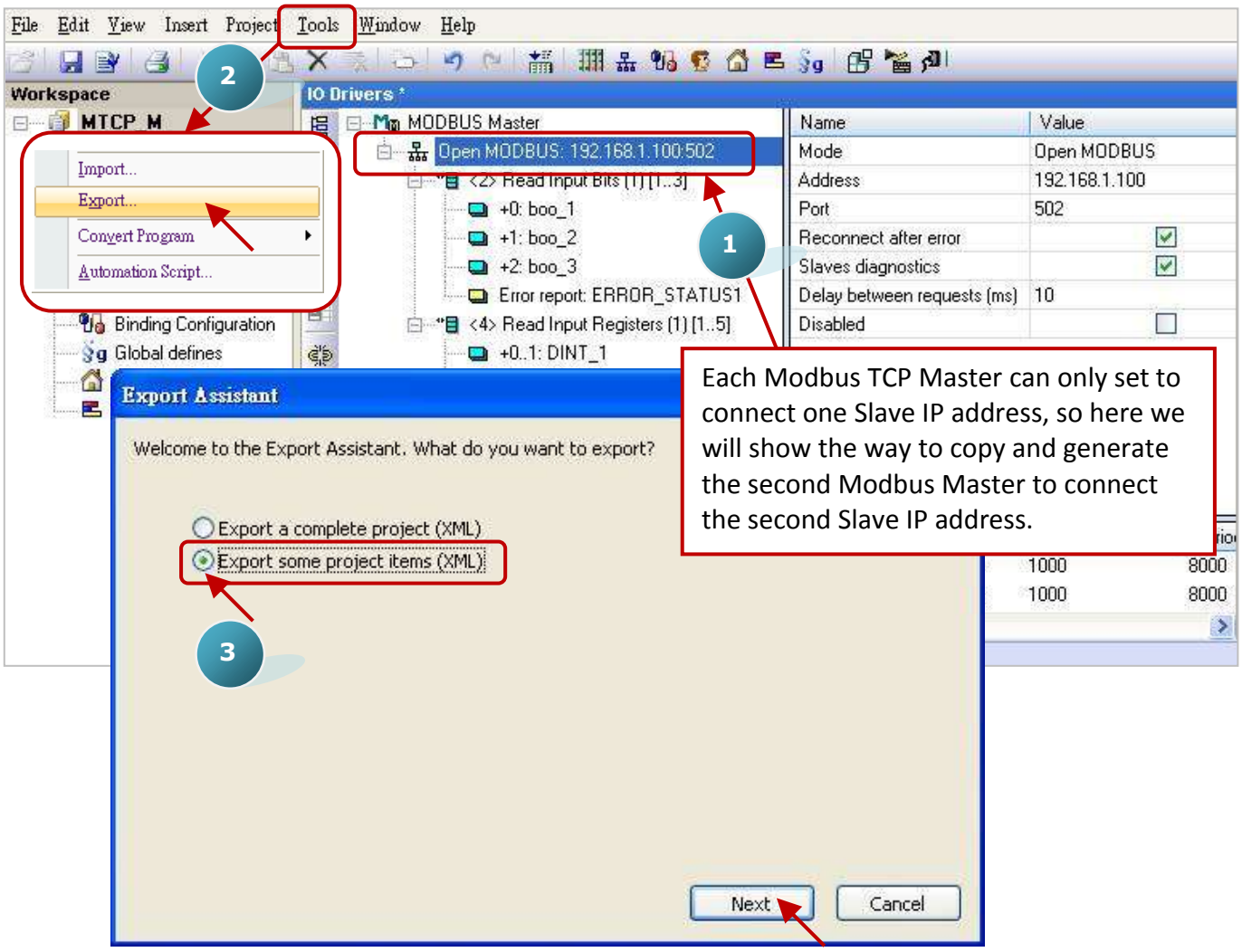
The previous section lists the way to enable the Win-GRAF PAC as a Modbus TCP Master device, and lists the way to read/write Modbus TCP Slave device. This section will list the way to create the redundant "Modbus Master Request", when one IP of the Modbus TCP Slave devices is disconnected, the other IP can still normally to be read/written data.

Application Diagram:

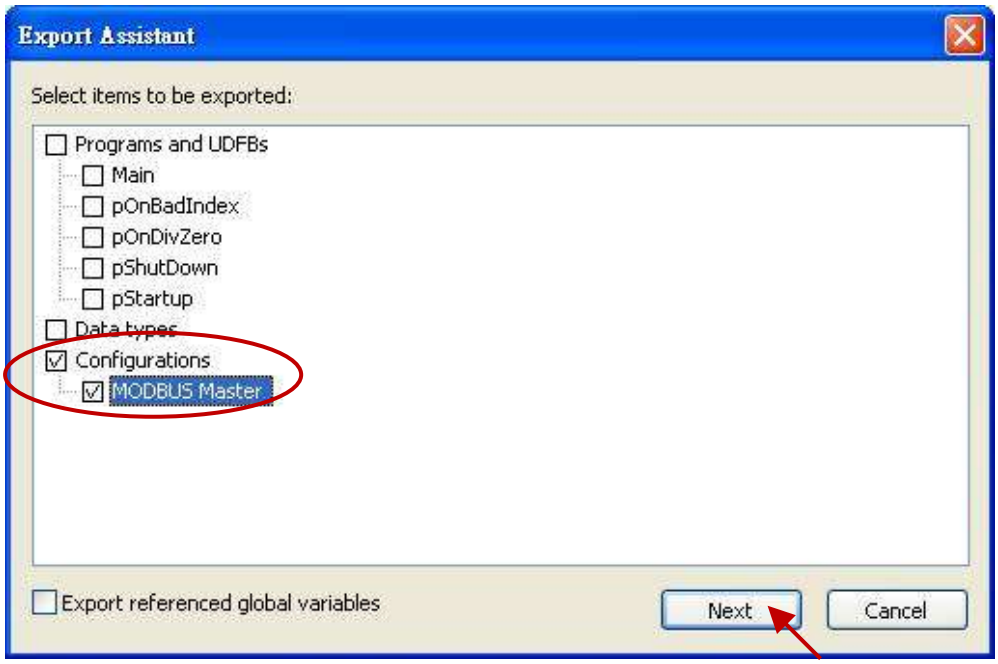


Follow The Steps:

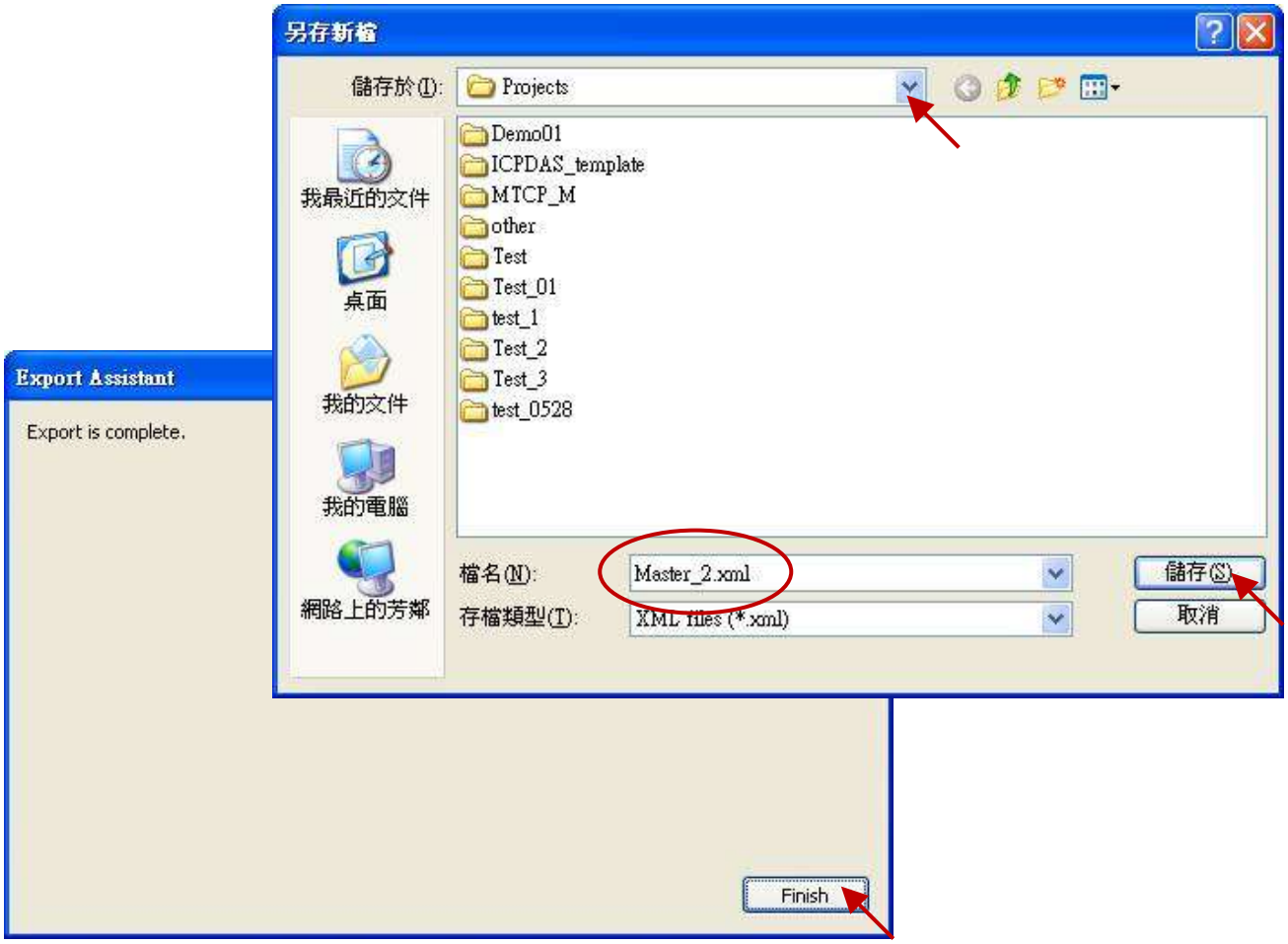
1. Click the "Open MODBUS:", and then click the menu bar "Tools" > "Export".
2. In the "Export Assistant" window, click "Export some project items (XNL)" and "Next".



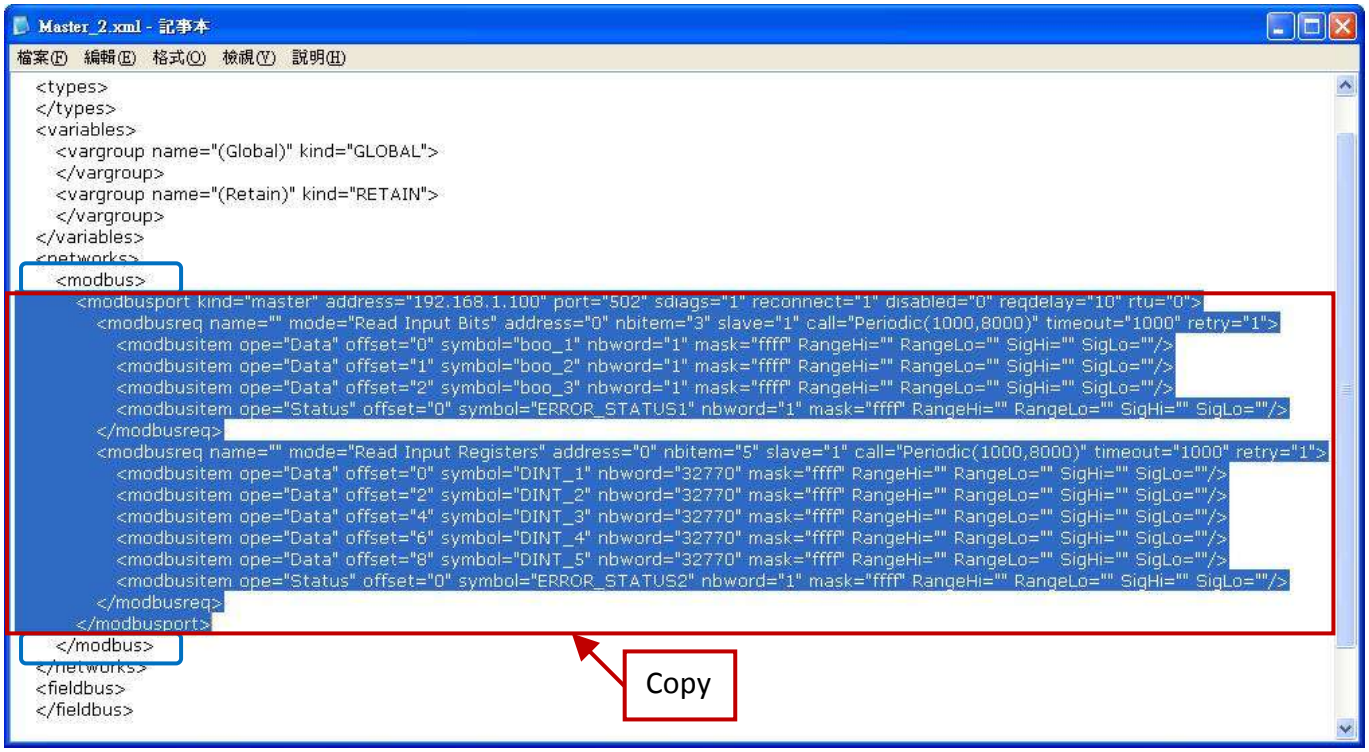
3. Check the “Configurations” and uncheck all other items, and then click “Next”.



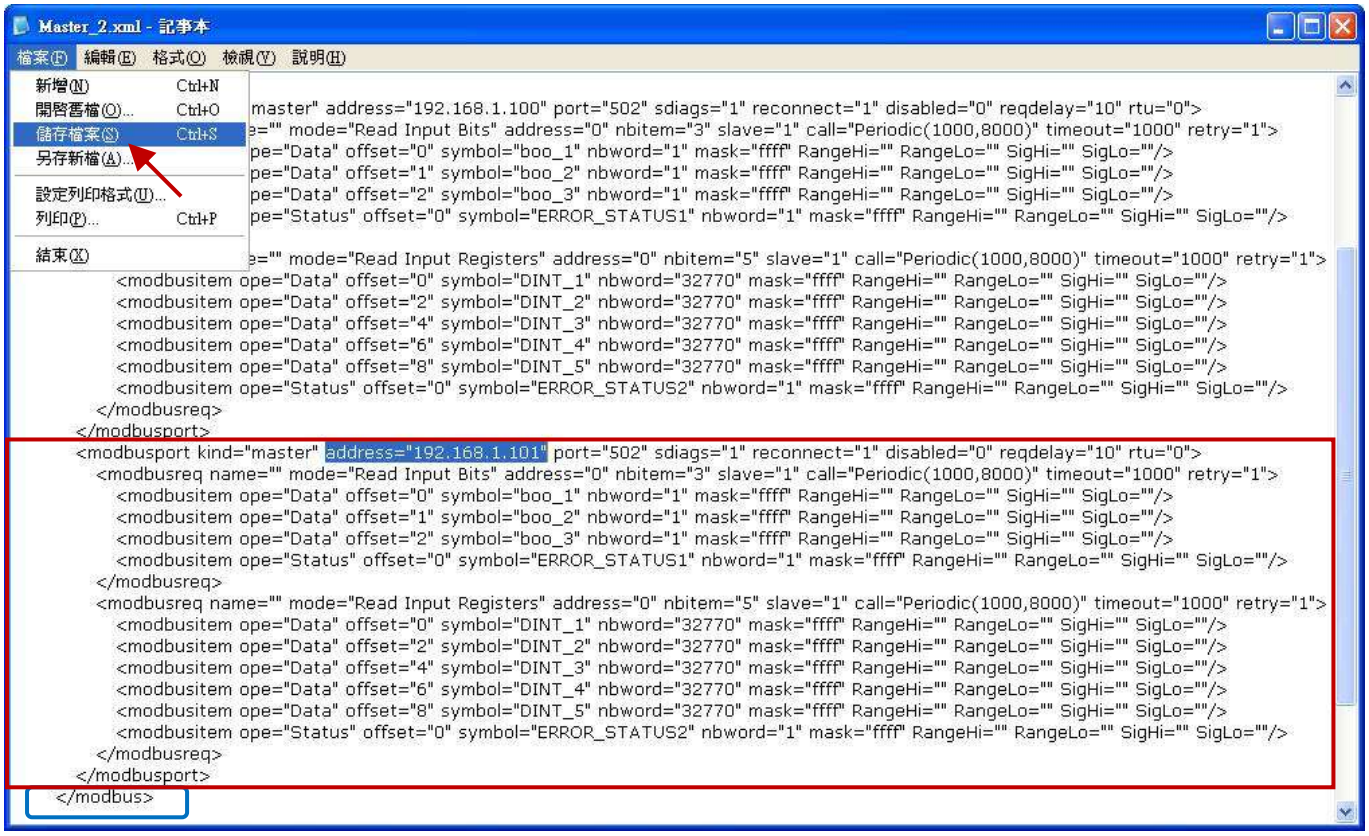
4. Browse a suitable path (default in C:\Win-GRAF\Projects) and named for the file (e.g., Master_2.xml), and then click "Save" button. Finally, click "Finish" to export the settings.



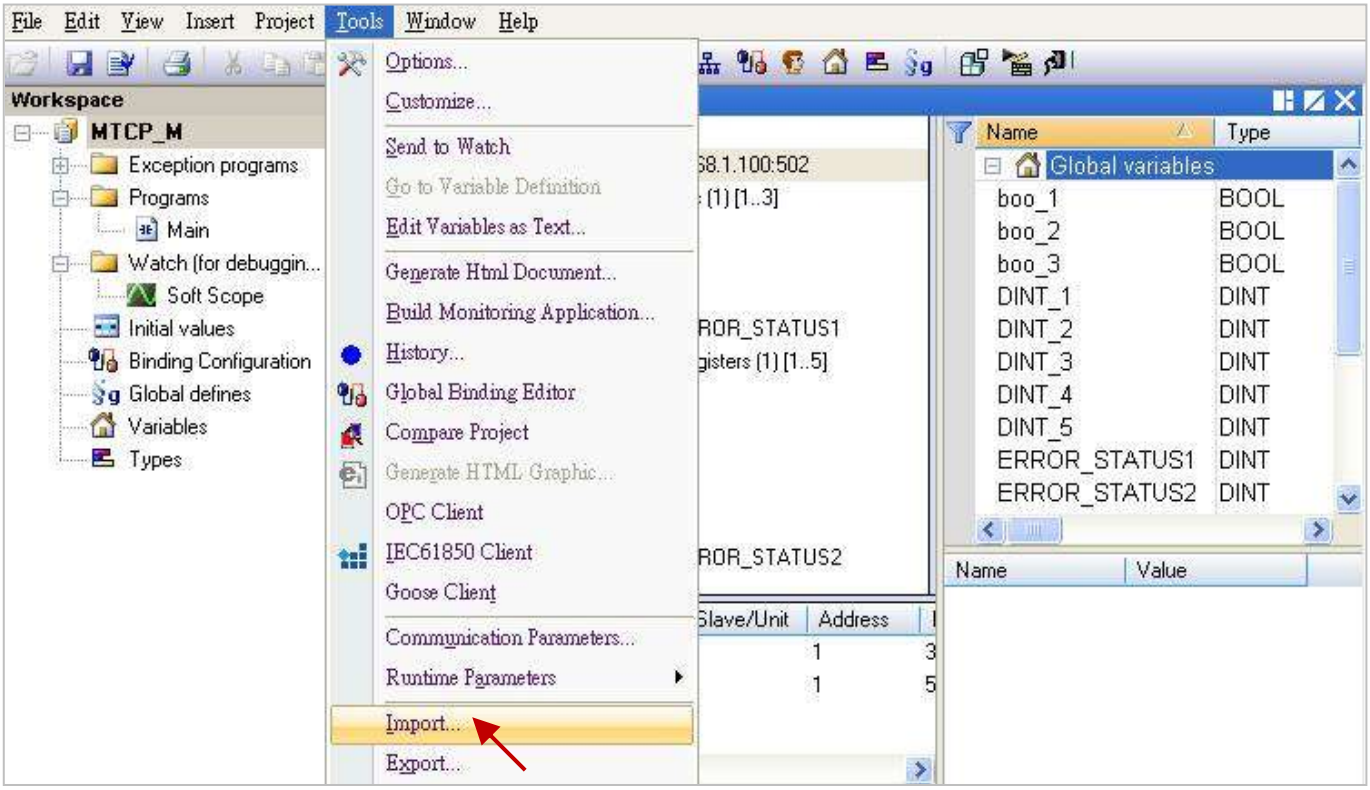
5. Using the Notepad software to open the file ".xml" that exported in the step 4, and then copy the content between the <modbus> and </modbus>.



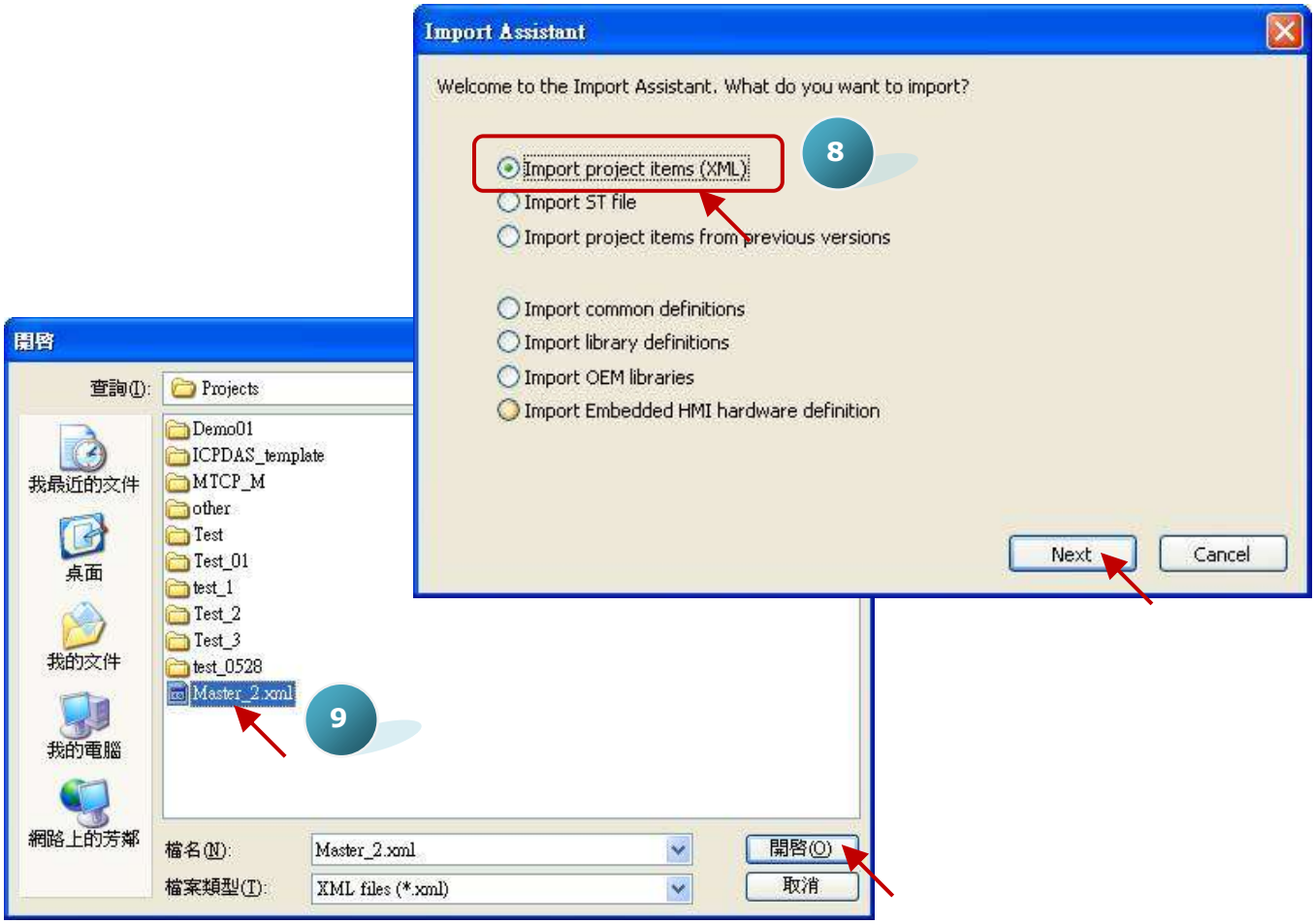
6. Paste the copied content above the </modbus>, and change the address to the second IP address of the Modbus Slave device (e.g., "192.168.1.101"), then save and close the file.



7. Click the Win-GRAF menu bar “Tools” > “Import”.

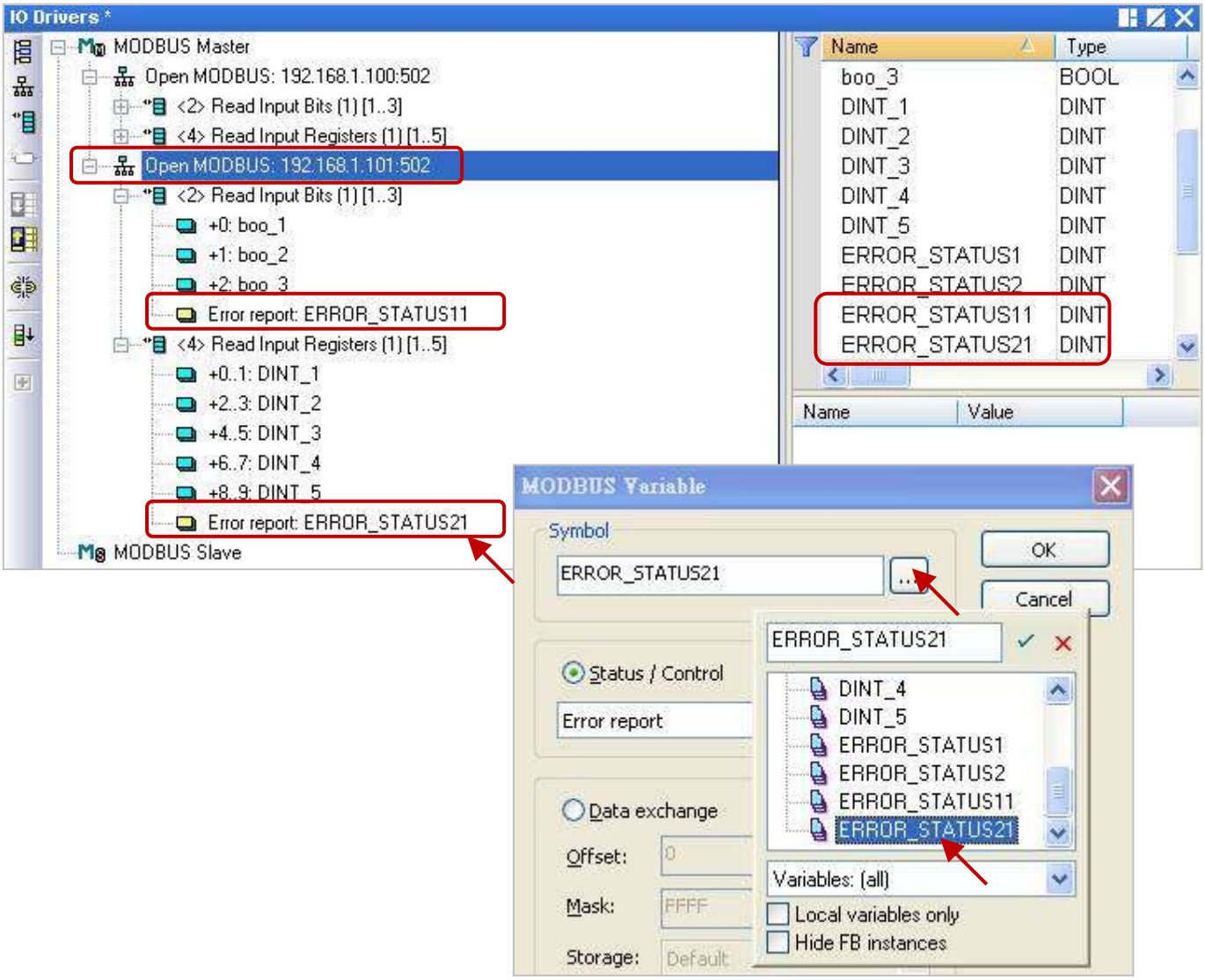


8. In the “Import Assistant” window, click “Import project items (XNL)” and “Next”.
9. Select the file you want to import (e.g., “Master_2.xml”) and click “Open” button.



10. Click “Finish” to finish the import action.

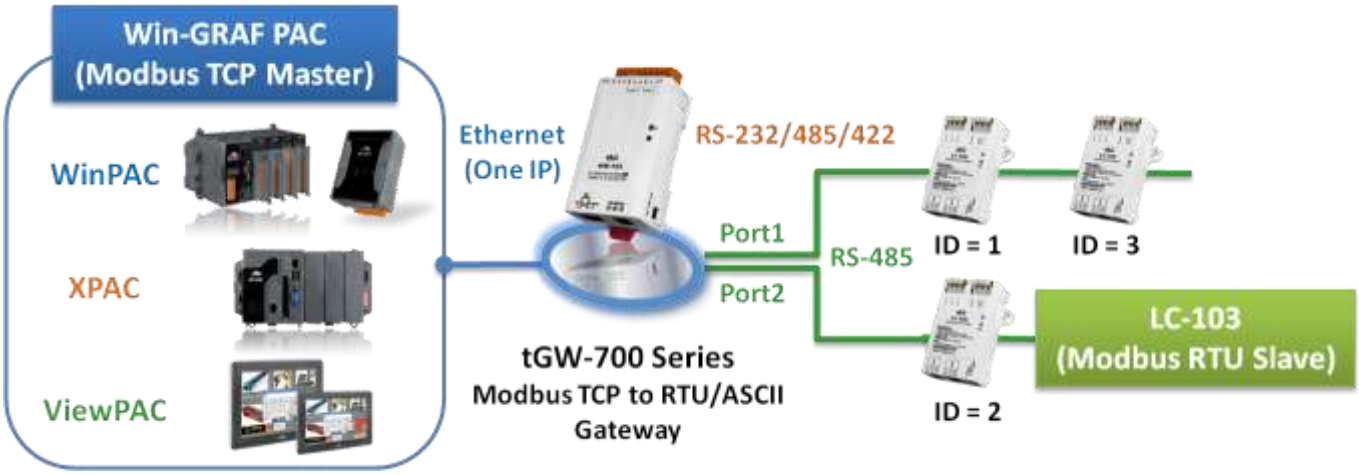
11. In the “IO Drivers” window, there has added a new “Open MODBUS” setting item that includes two “Modbus Master Request” for using to read the DI and AI. One “Error report” is used to check the IP connecting status, so please add two “DINT” variables in the Variable area (e.g., “ERROR_STATUS11”, “ERROR_STATUS21”) and double click the “Error report” to assign variables.



5.4 Connecting the tGW-700 to Expand Modbus RTU Master Ports

If using the Modbus RTU (RS-232/485/422) device to transmit data in a long-distance application area, the user will normally choose a lower baud rate for better signal quality. But, using this way will cause low transmission efficiencies. In order to improve this problem, ICP DAS releases the tGW-700 series products (tiny Modbus TCP to RTU/ASCII gateway) for converting Ethernet/RS-485 signals so that the user can reduce the RS-485 cable lengths and solve the issue with inefficient communications.

This section will provide a demo program (demo_tgw725.zip) to describe how the Win-GRAF PAC communicates with LC-103 modules via the tGW-700 gateway (as the figure below).



5.4.1 Using the tGW-700 Series (Modbus TCP to Modbus RTU/ASCII Gateway)

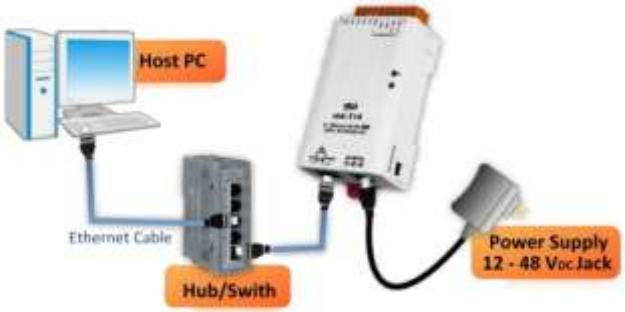
The tGW-700 module is a Modbus TCP to RTU/ASCII gateway that enables a Modbus TCP host (e.g., WP-8xx8) to communicate with serial Modbus RTU/ASCII devices through an Ethernet network, and eliminates the cable length limitation of legacy serial communication devices. Visit the tGW-700 series webpage for more information on
http://www.icpdas.com/root/product/solutions/industrial_communication/pds/tgw-700.html

tGW-700 series User Manual
http://ftp.icpdas.com/pub/cd/tinymodules/napdos/tgw-700/document/ (See the chapter 3 & chapter 4 to know the way of network setting, testing and web function configuration for the tGW-700 module.)

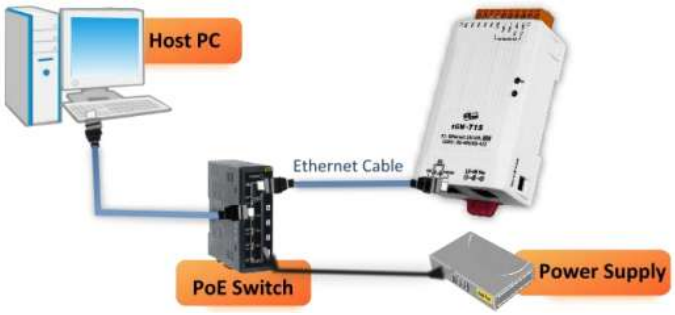
Before using the tGW-700, the user must configure its network and COM Port setting:

- **Connect the Power Supply and the Host PC**
  1. Check that the Init/Run switch is in the “Run” position.

2. Connect both the tGW-700 and the Host computer to the same sub-network or the same Ethernet Switch, and then power on the tGW-700.

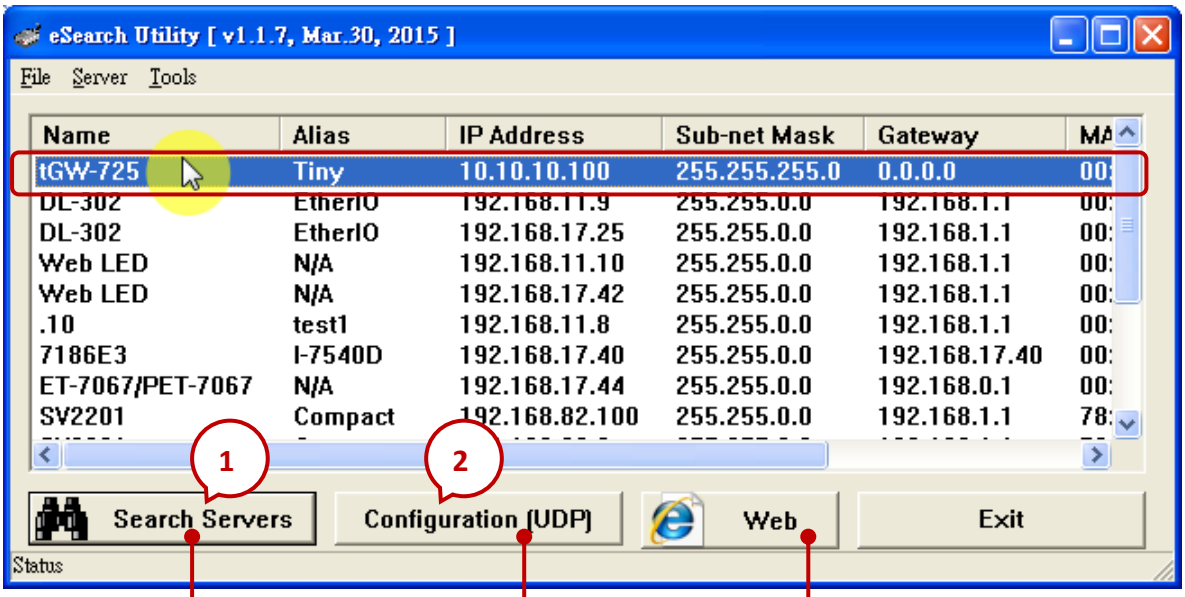


+12 to +48 V_{DC} jack Power Supply (Non-PoE)



PoE Power Supply

- **Install the "eSearch Utility", and then Search and Configure the Network Setting for the tGW-700**
  http://ftp.icpdas.com/pub/cd/tinymodules/napdos/software/modbus_utility/

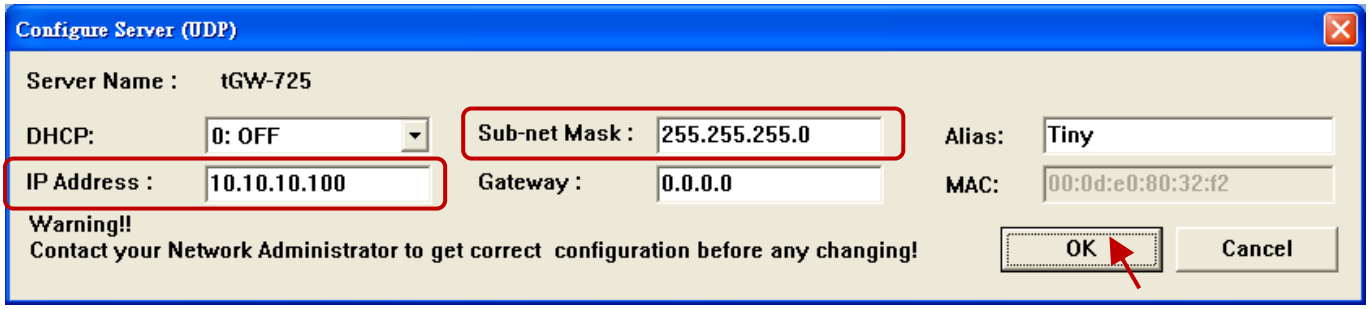


Search your tGW-700.

Set the tGW-700's IP / Mask / Gateway. (Contact your Network Administrator to get correct configuration)

Open the tGW-700 Web Server. (**Note:** Both the tGW-700's and PC's IP addresses must on the same sub-network. See chapter 4 of the tGW-700 user manual.)

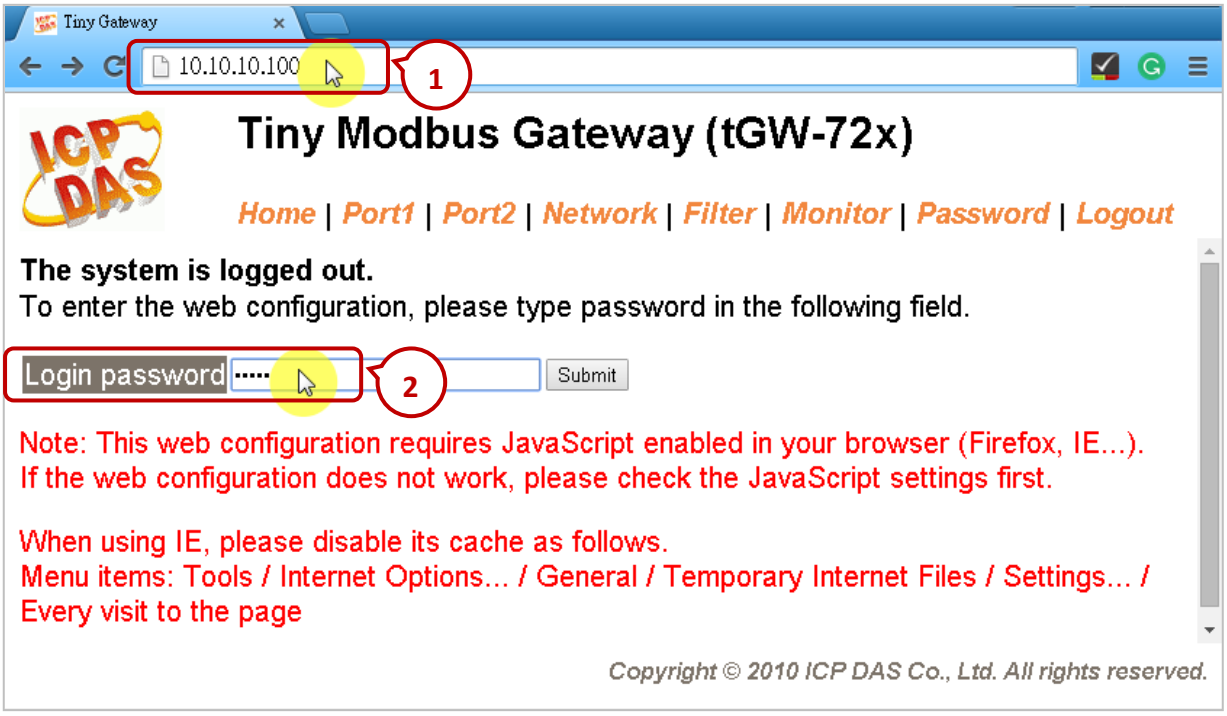
Please contact your Network Administrator to get the correct IP, Mask and Gateway addresses. After completing these settings, click the "OK" button and they will take effect within 2 seconds.



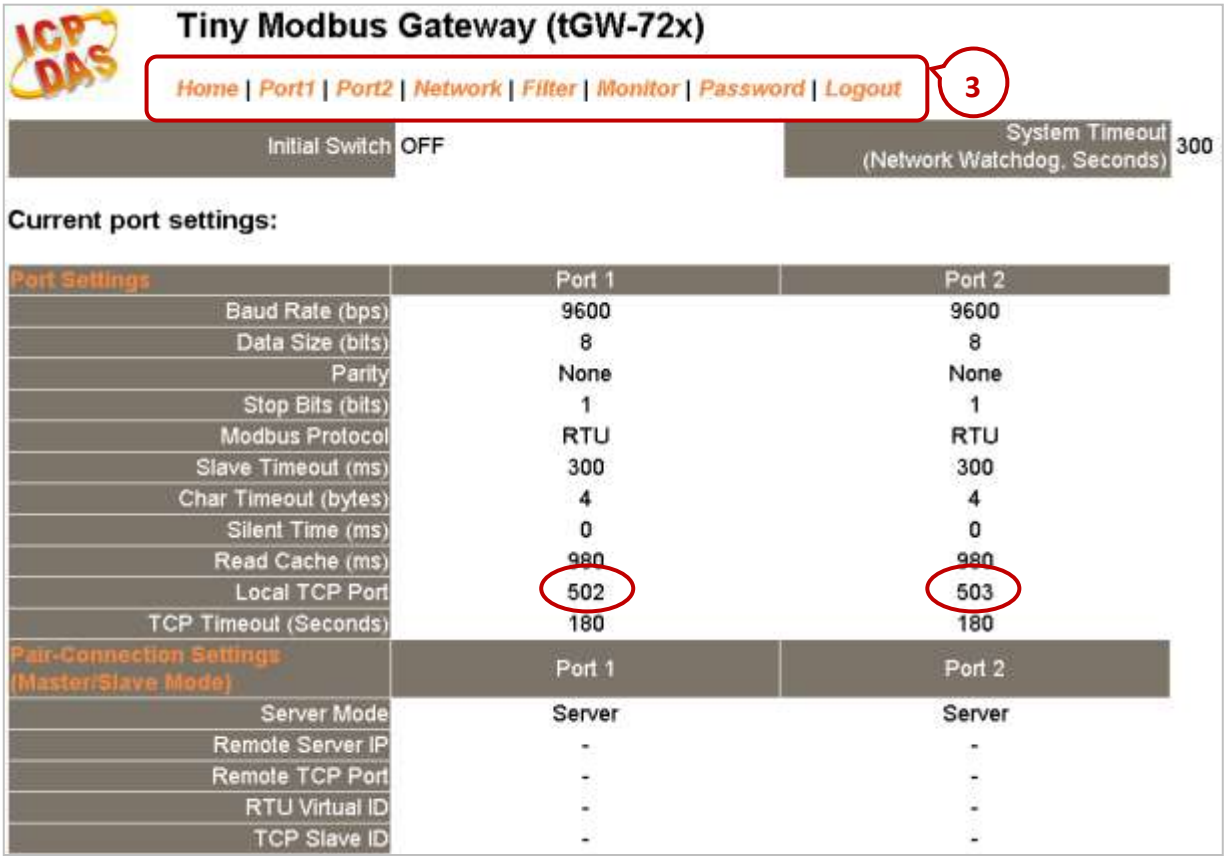
- **Web Configuration**

You can refer the tGW-700 user manual (chapter 4) to view the configuration way for all features. The following will describe the COM Port setting.

1. Enter the tGW-700's IP address on the web browser. (**Note:** It must on the same sub-network with your PC's IP).
2. Enter the password (the factory default password is "admin").



3. After logging in, the main page (Home) will display the current port setting. The user can also click "Port1" or "Port2" tab to modify the settings.



5.4.2 Connecting the tGW-700 Series and the LC-103 module (1 DI, 3 Relay)

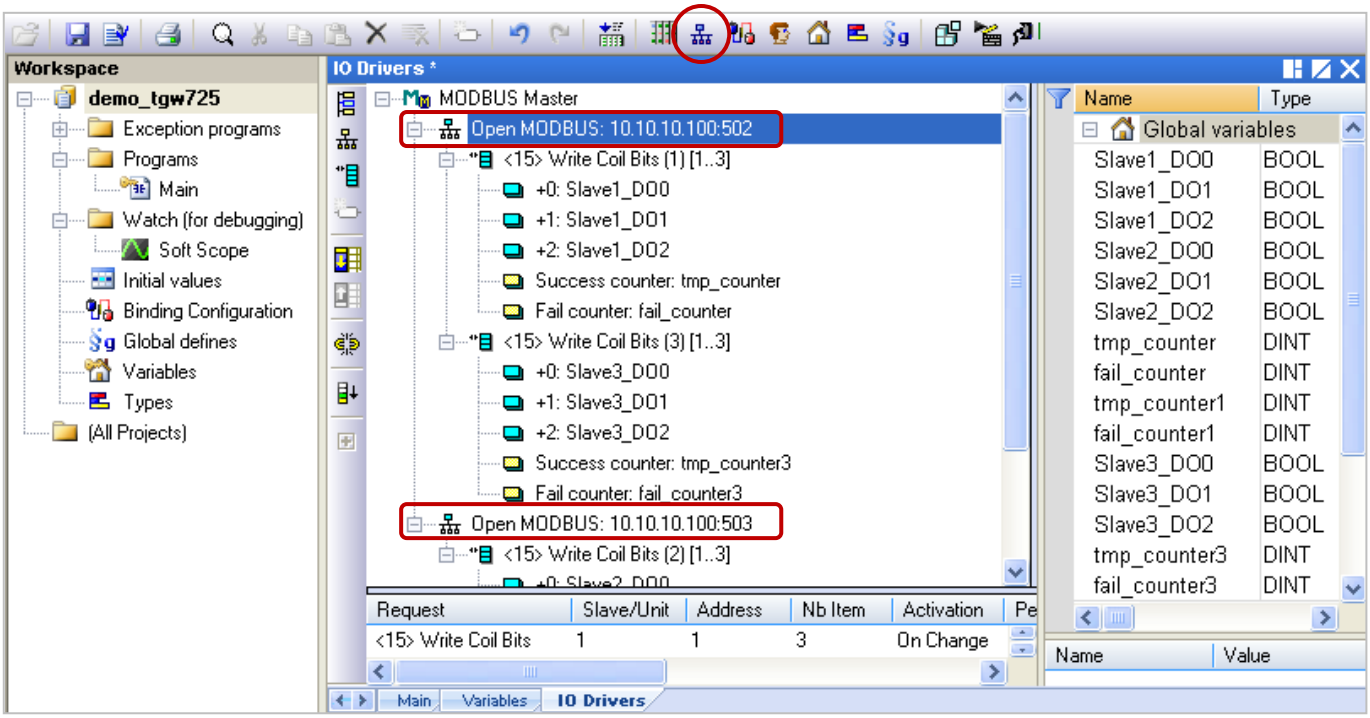
In this section, we provide a demo project (demo_tgw725.zip) to describe how the Win-GRAF PAC communicates with LC-103 modules via the tGW-725 (the Modbus TCP to Modbus RTU/ASCII gateway with two RS-285 ports). You can run the Win-GRAF Workbench and click "File → Add Existing Project → From Zip..." to open this project in the Win-GRAF PAC CD (CD-ROM: \Napdos\Win-GRAF\demo-project).

The LC-103 is an easy-to-use lighting control module that supports the Modbus RTU protocol and provides 1 channel for digital input and 3 channels for relay output. Before using this module, set its ID No. depends on your application needs, for example, if the required ID is "1", simply adjust the rotary switch to "1" at the bottom of the module. Visit the LC-103 webpage for more detailed information: http://www.icpdas.com/root/product/solutions/remote_io/rs-485/lighting_control/lc-103.html

The LC series module user manual: ftp://ftp.icpdas.com.tw/pub/cd/8000cd/napdos/lc/

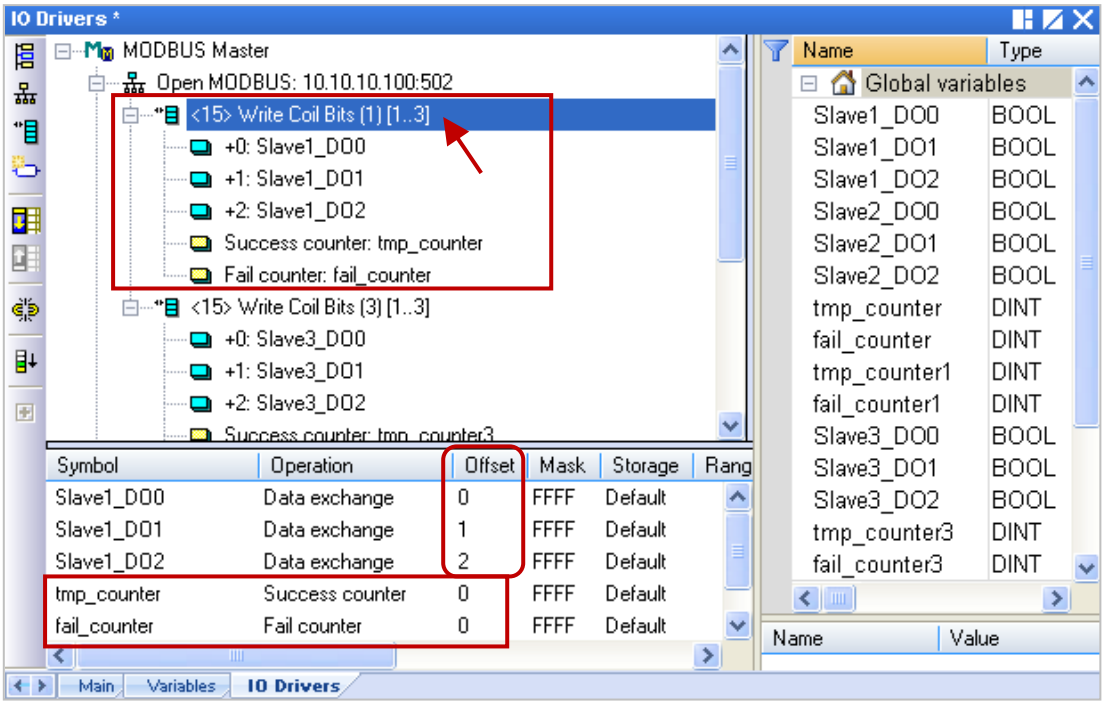
Demo Project Description: (Refer the Section 5.2 to know how to create this project.)

1. Click the "Open Fieldbus Configuration" tool button to open the "IO Drivers" window.

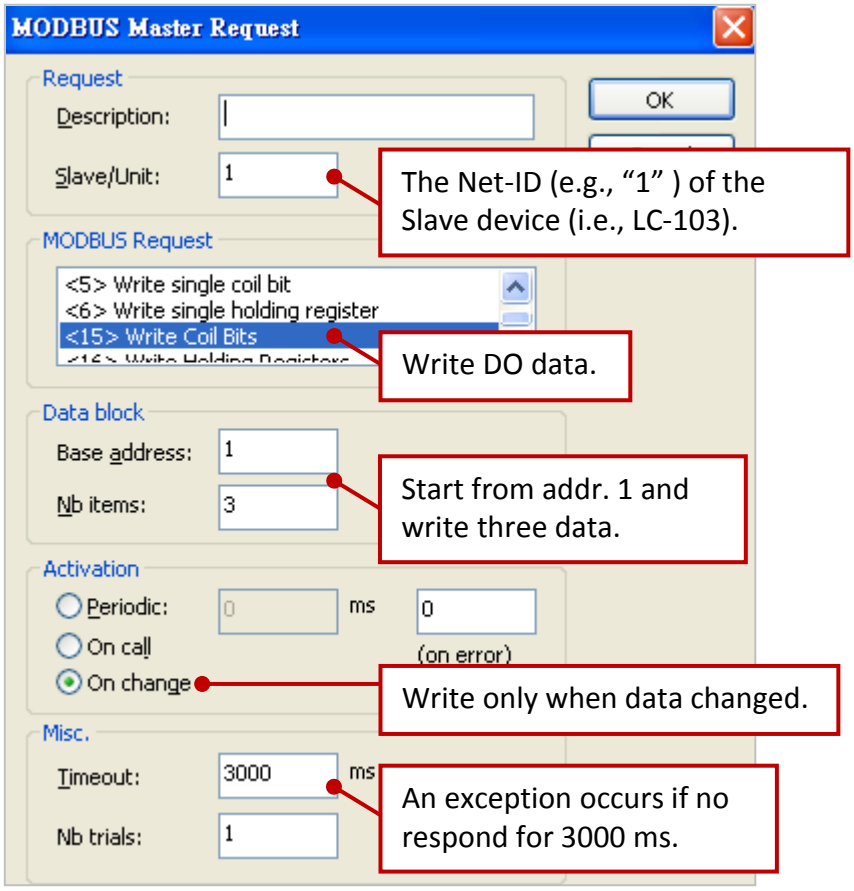


On the screen, the "Open MODBUS: 10.10.10.100:**502** / Open MODBUS: 10.10.10.100:**503**" means that the tGW-725's IP address is "10.10.10.100" and using two COM ports (RS-485) No. - "502" and "503". And, there are two LC-103 modules (Slave ID = "1" and "3") connected to its COM1 and one LC-103 connected to the COM2 (Slave ID = 2). The following will describe the configuration way of each Modbus Master Request one-by-one.

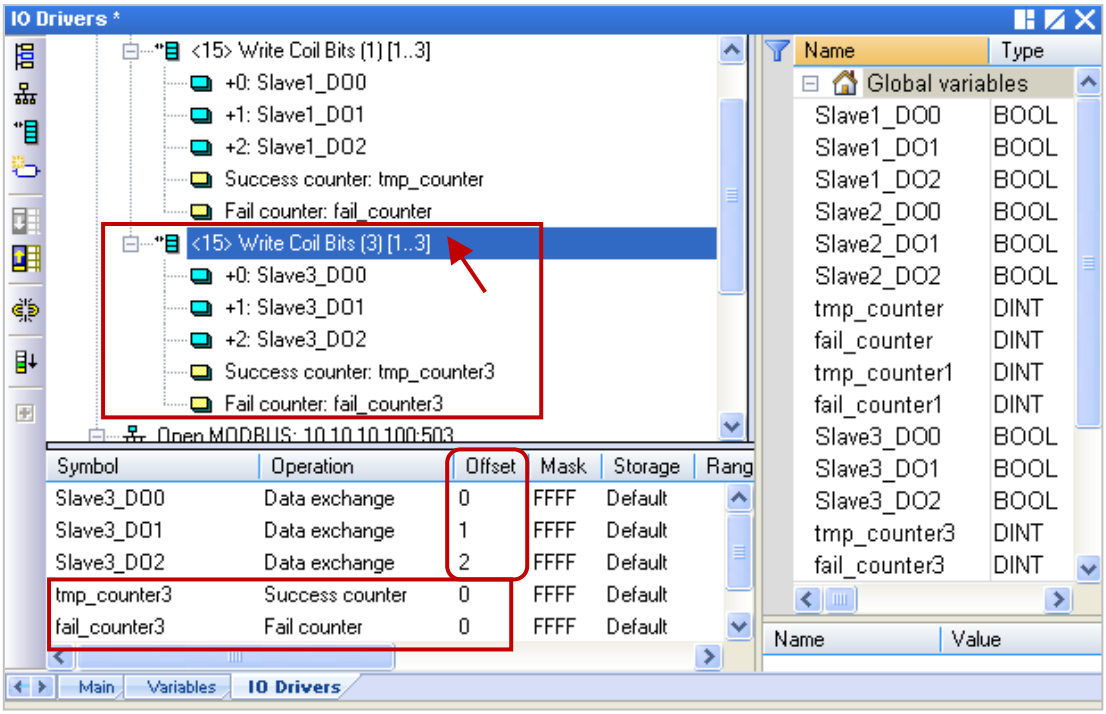
2. Mouse double-click the first data block under the COM1 (Port = 502) to view this Modbus Master request. In this example, the Win-GRAF PAC (Modbus TCP Master) send three DO commands to the **LC-103 (Slave ID = 1)** via the tGW-725's **COM1 (Port = 502)**. As the figure below, the "Operation" is set to "Success counter" (or "Fail counter") that means this variable value will add 1 if the command was successfully sent (or failed). Moreover, the "Offset" value of these variables must set as "0".



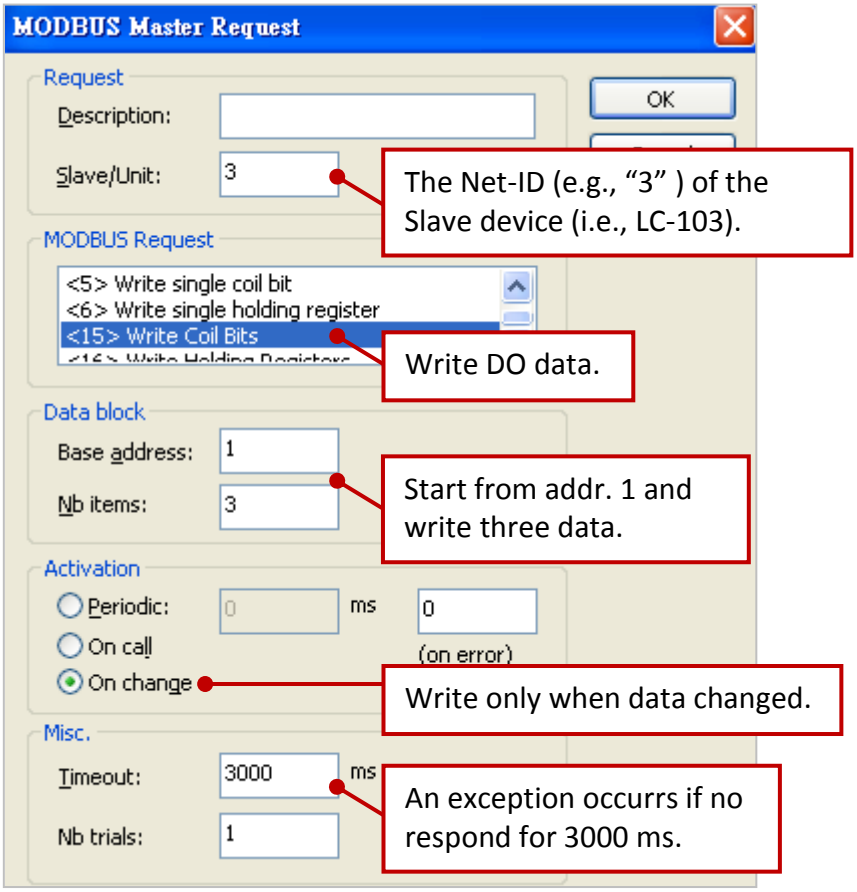
Note: The "Offset" value starts at "0" and the Modbus address of variable is equal to the "Offset" value plus 1 (Base address).



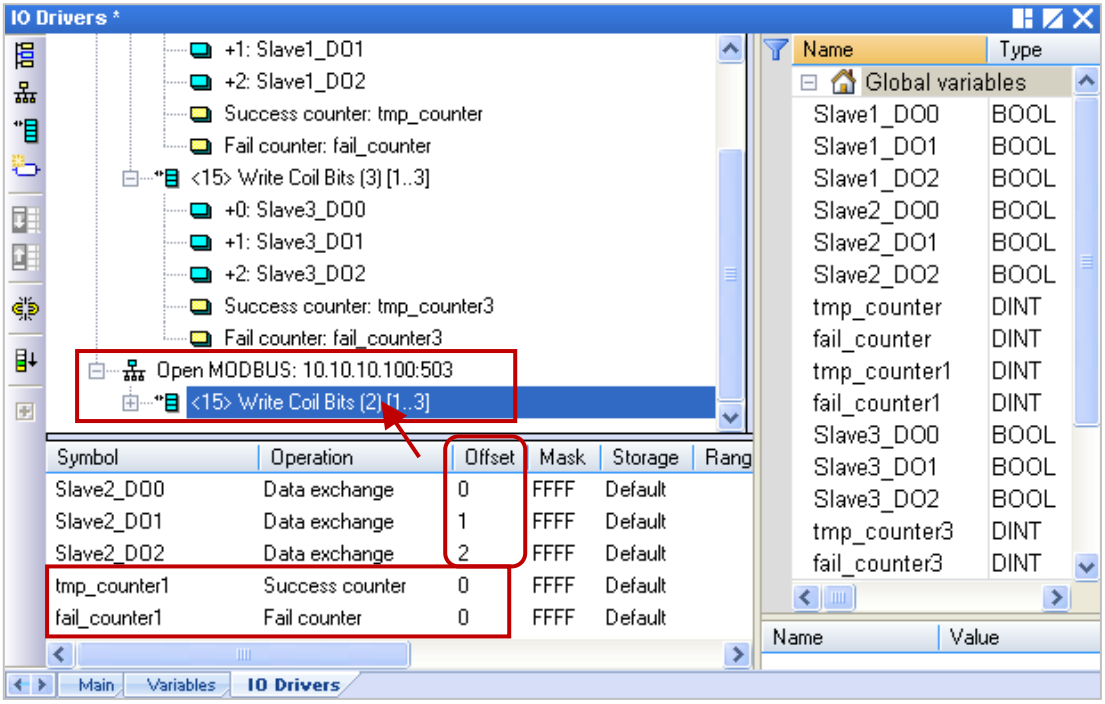
3. Mouse double-click the 2nd data block under the COM1 (Port = 502) to view this Modbus Master request. In this example, the Win-GRAF PAC (Modbus TCP Master) send three DO commands to the **LC-103 (Slave ID = 3)** via the tGW-725's **COM1 (Port = 502)**. As the figure below, the "Operation" is set to "Success counter" (or "Fail counter") that means this variable value will add 1 if the command was successfully sent (or failed). Moreover, the "Offset" value of these variables must set as "0".



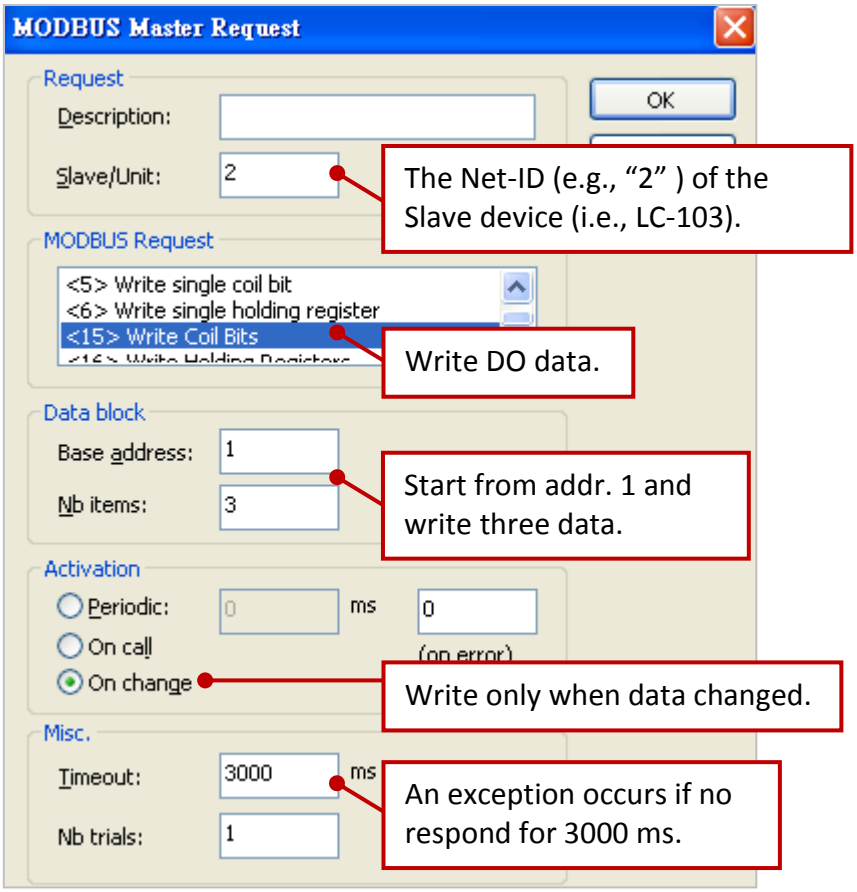
Note: The "Offset" value starts at "0" and the Modbus address of variable is equal to the "Offset" value plus 1 (Base address).



4. Mouse double-click the data block under the COM2 (Port = 503) to view this Modbus Master request. In this example, the Win-GRAF PAC (Modbus TCP Master) send three DO commands to the **LC-103 (Slave ID = 2)** via the tGW-725's **COM2 (Port = 503)**. As the figure below, the "Operation" is set to "Success counter" (or "Fail counter") that means this variable value will add 1 if the command was successfully sent (or failed). Moreover, the "Offset" value of these variables must set as "0".



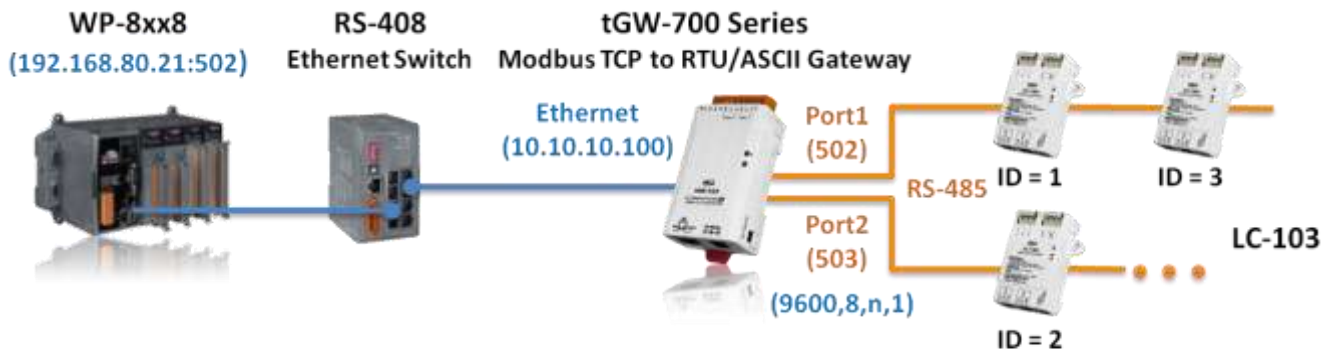
Note: The "Offset" value starts at "0" and the Modbus address of variable is equal to the "Offset" value plus 1 (Base address).



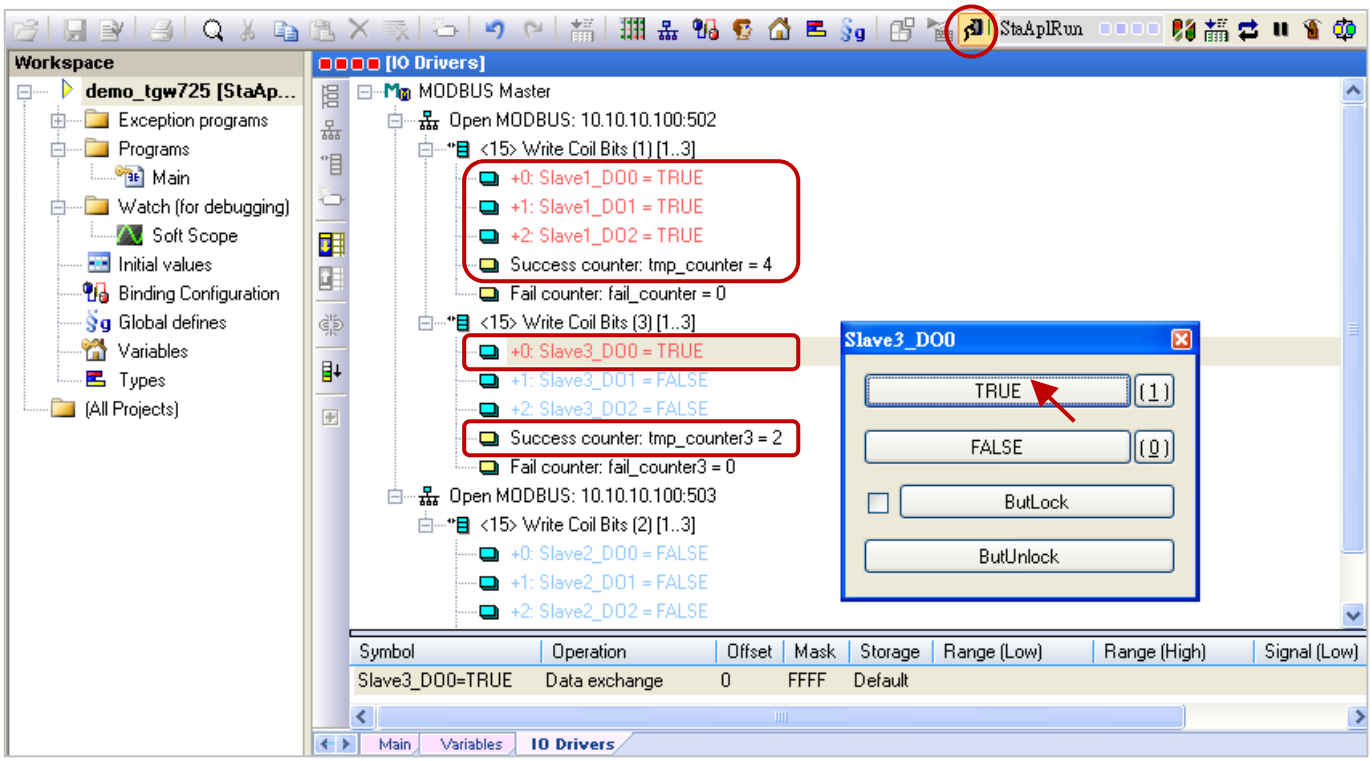
5.4.3 Test the Demo Project (demo_tgw725.zip)

Before testing this demo project, download it to your Win-GRAF PAC. (If you're not familiar with it, refer the Section 2.3.4 and Section 2.3.5.)

The Hardware Wiring:



After connecting with the Win-GRAF PAC, double-click on any DO variable and set it as "TRUE" in the "I/O Drivers" window. If the write operation is successful, then the "tmp_counter" value will add "1".



Note: When the Win-GRAF PAC boots up, it will send the Modbus request to the Modbus Slave device. So, you can see the "tmp_counter" value starts at "1" which means this data write is successful.

Chapter 6 Retain Variable and Data Storage

6.1 Retain Variable

This chapter lists the way to use the "RETAIN_VAR", "RETAIN_ARY", "RETAIN_FLAG_GET", "RETAIN_FLAG_SET" and "RETAIN_FLAG_CLR" Functions. The Win-GRAF PACs are built-in the Retain memory for users to store the retain variable data that will not lose due to the PAC shutdown and can retain the last value at the next time reboot.

In the shipping CD (\Napdos\Win-GRAF\demo-project), you can find the demo project for this chapter (demo_retain.zip), please refer Chapter 12 to restore this project (Execute File> Add Existing Project > From Zip) and set the current IP address of your PAC.

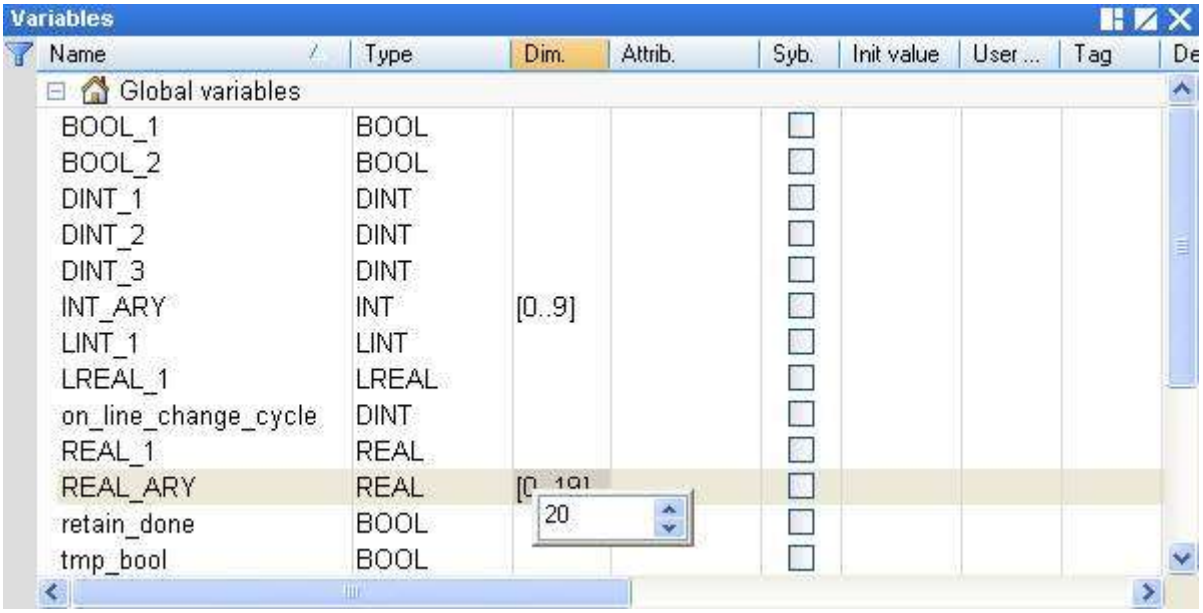
Note: Function "Retain_Var()" or Retain_Ary() can only be used in the first PAC Cycle or in the Cycle that performs the On-line Change. If use them in other Cycle, it will return "FALSE". If the Retain Variable has not assigned any initial value and the PAC calls the Function, the return value is not meaningful; users need to assign appropriate initial values to all Retain Variables at least once.

ST Program:

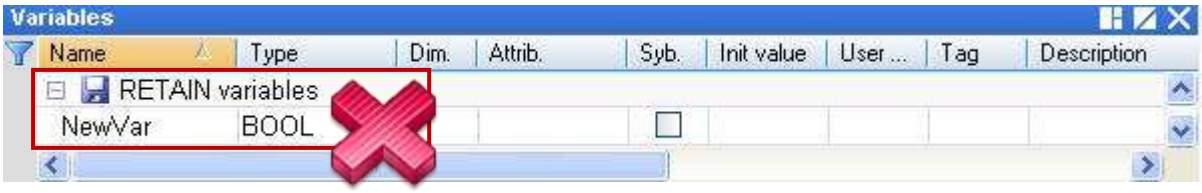
This demo uses Function Retain_Var() and Retain_Ary().

```
(* "on_line_change_cycle" is declared as DINT
                                (nonezero means it is in the cycle just after doing on line change).
  "retain_done" is declared as BOOL and inited as FALSE.
  "tmp_bool" is declared as BOOL.
 *)

 on_line_change_cycle  :=  GetSysInfo (_SYSINFO_CHANGE_CYCLE) ;
 if (retain_done = FALSE) or (on_line_change_cycle <> 0) then
      retain_done := TRUE ;  (* just do it one time *)
      tmp_bool := Retain_Var ( DINT_1 , 1) ; (* retain a DINT variable *)
      tmp_bool := Retain_Var ( DINT_2 , 2) ;
      tmp_bool := Retain_Var ( REAL_1 , 3) ; (* retain a REAL variable *)
      tmp_bool := Retain_Var ( BOOL_1 , 4) ; (* retain a BOOL variable *)
      tmp_bool := Retain_Var ( BOOL_2 , 5) ;

     (* retain 10 elements of an INT array variable at retain addr starting at 6. *)
      tmp_bool := Retain_Ary ( INT_ARY ,  6 ,  10) ;

     (* retain 20 elements of a REAL array variable at retain addr starting at 16. *)
      tmp_bool := Retain_Ary ( REAL_ARY , 16 , 20) ;

      tmp_bool := Retain_Var ( DINT_3 , 36) ;
      (* 64-bit variable can use only addr from 10,001 to 12,000 *)
      tmp_bool := Retain_Var ( LINT_1  , 10001) ;  (* retain a LINT variable (64-bit) *)
      tmp_bool := Retain_Var ( LREAL_1 , 10002) ; (* retain a LREAL variable (64-bit) *)
    end_if ;
```

You can check/set variables in the "Variables" window, if you are not familiar with the way of the variable declaration, please refer Section 2.2.2 and Section 2.3.1.



Note: ICP DAS Win-GRAF PAC does not support the "RETAIN variables" Functions in the "Variables" window, so please refer the five Functions in the following sections to use the Retain Variables.



6.1.1 RETAIN_VAR (Retain a Variable)

Tips:
Press "F1" key to see more details.

Name:
A variable name (DO NOT use Array variable or String).
Variable type can be BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, LINT or LREAL.

Addr:
Data Type: DINT. The address number for retaining the variable, can be 1 to 12,000.

Q:
Data Type: BOOL. TRUE: Ok; FALSE: Error.

Note:

1. One Addr can accept only one variable (or one element of the array).
   DO NOT assign the same Addr to two variables (or more), or the Retain Value will be wrong.

2. 64-bit data type (LINT or LREAL) can use only the Addr No. from 10,001 to 12,000.

3. Other data type (BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL or TIME) can use the Addr No. from 1 to 12,000.

6.1.2 RETAIN_ARY (Retain an Array Variable)

Tips:
Press “F1” key to see more details.

Name[]:
An ARRAY variable name (DO NOT use String).
Variable type can be BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, LINT or LREAL.

Addr:
Data Type: DINT. The starting address number for retaining the array variable; can be 1 to 12,000.

Num:
Data Type: DINT. The amount of elements in the Array variable to be retained.
For example:
If there are 100 elements in an Array variable, set "Num" to "1 to 100" is correct, but if set it more than 100 that is not correct.
If there are 5 elements in an Array variable, set "Num" to "1 to 5" is correct, but if set it more than 5 that is not correct.

Q:
Data Type: BOOL. TRUE: Ok; FALSE: Error.

Note:

1. One Addr can accept only one variable (or one element of array).
   DO NOT assign the same Addr to two variables (or more), or the Retain Value will be wrong.

2. 64-bit data type (LINT or LREAL) can use only the Addr No. from 10,001 to 12,000.

3. Other data type (BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL or TIME) can use the Addr No. from 1 to 12,000.

6.1.3 RETAIN_FLAG_SET/GET/CLR (Set/Get/Clear the Retain Flag)

How to Use:

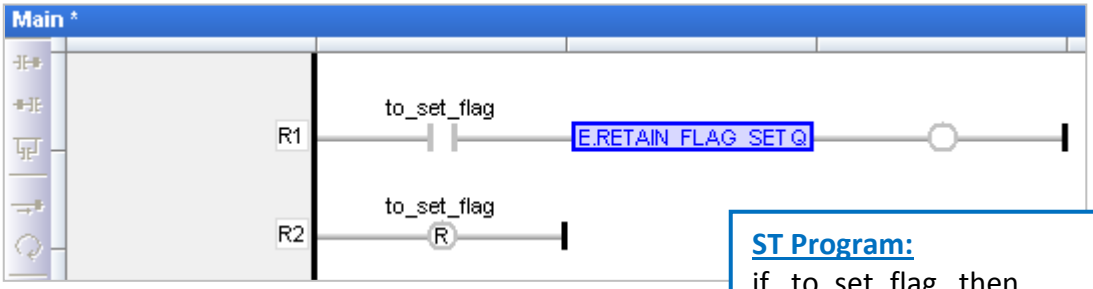
The "Retain Flag" is a flag (TRUE/FALSE) stored by users in the retain memory. Users can set this retain flag to indicate "All retain data has been assigned a proper value before". When a PAC starts without setting a proper value to retain variable before, the data of the retain variable read from the retain memory is not correct (it is normally a strange value). So users have to assign proper value to all retain variable at least once to let the application work well. Then after user can call the "Retain_Flag_Set()" to set the retain flag. It means "All retain data has been assigned a proper value".
To get the state of the Retain Flag, please call "Retain_Flag_Get()".
To clear the state of the Retain Flag, please call "Retain_Flag_Clr()".

ST Program:

```
(*  "on_line_change_cycle" is declared as DINT
    (nonezero means it is in the cycle just after doing on line change) .
   "retain_done" is declared as BOOL and inited as FALSE .
   "tmp_bool", "retain_flag" and "to_set_flag"  are declared as BOOL.
  *)

  on_line_change_cycle  :=  GetSysInfo (_SYSINFO_CHANGE_CYCLE) ;

  if (retain_done = FALSE) or (on_line_change_cycle <> 0) then
    retain_done := TRUE ;  (* just do it one time *)
    tmp_bool := Retain_Var( DINT_1 , 1) ;  (* retain a DINT variable *)
    tmp_bool := Retain_Var( DINT_2 , 2) ;
    tmp_bool := Retain_Var( REAL_1 , 3) ; (* retain a REAL variable *)
    tmp_bool := Retain_Var( BOOL_1 , 4) ; (* retain a BOOL variable *)

    (* ... After doing all the Retain Functions ... *)
    retain_flag := Retain_Flag_Get();
    if  (retain_flag = FALSE)  then
      (*If Retain variable does not set up any proper value, you can do some proper operation here. *)
      (* ... *)
    end_if ;

  end_if ;

   (* When all Retain variables are assigned proper values,
      remember to set the "to_set_flag" to "TRUE" for calling "Retain_Flag_Set() once,
      so that, when next time you use the "Retain_Flag_Get()", it can return "TRUE".
      *)
  if  (to_set_flag = TRUE)  then
    to_set_flag := FALSE ;
    tmp_bool := Retain_Flag_Set() ;
  end_if ;
```

LD Program:
(Press "F1" key to see the detailed setting descriptions.**)**

RETAIN_FLAG_SET: Set the retain flag.



ST Program:

```
if  to_set_flag  then
    to_set_flag  :=  FALSE  ;
    TMP_BOOL  :=  Retain_Flag_Set() ;
end_if ;
```

Q: Data Type: BOOL. Always return TRUE.

RETAIN_FLAG_GET: Get the state of the retain flag.

ST Program:

```
retain_flag  :=  Retain_Flag_Get() ;
```

Q: Data Type: BOOL.
"TRUE": flag is set;
"FALSE": flag is not set.

RETAIN_FLAG_CLR: Clear the retain flag.

ST Program:

```
if  to_clr_flag  then
    to_clr_flag  :=  FALSE  ;
    TMP_BOOL  :=  Retain_Flag_Clr() ;
end_if ;
```

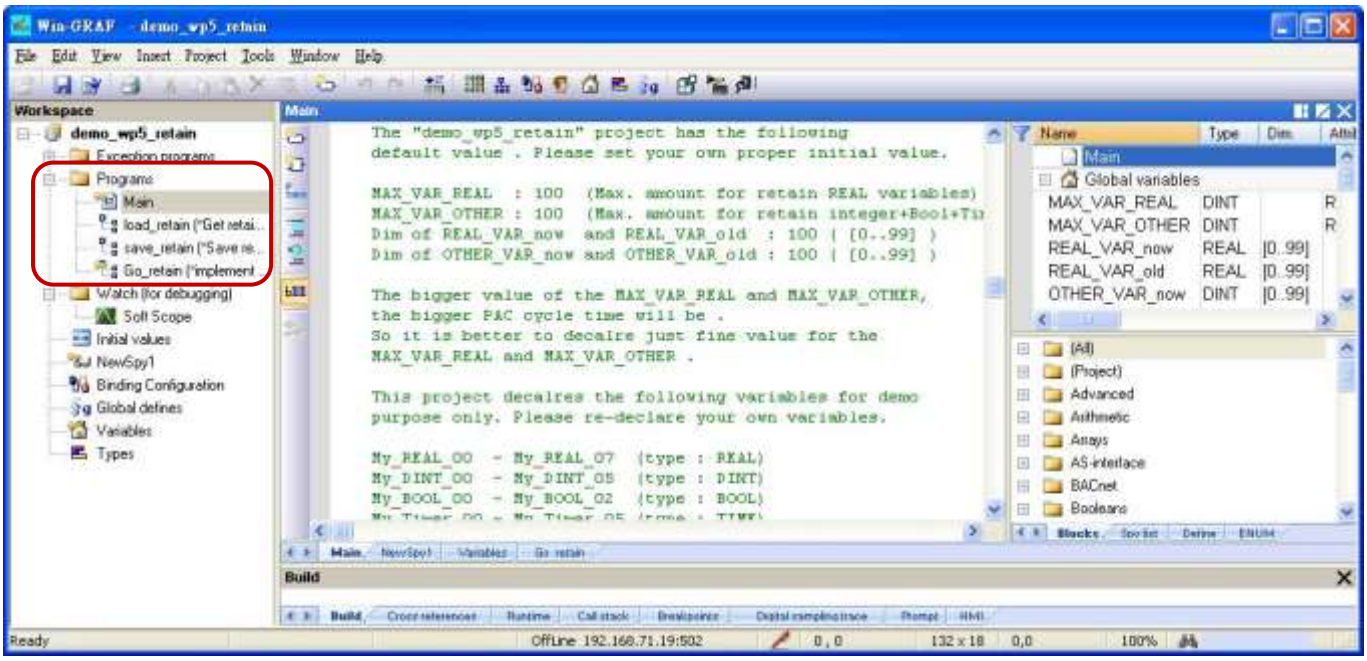
Q: Data Type: BOOL. Always return TRUE.

6.2 Retain Variable (Using files)

You can refer the following demo project (demo_wp5_retain.zip) to use retain variables with files in the \System_disk\Win-GRAF\ .

In the shipping CD (\Napdos\Win-GRAF\demo-project), you can find the demo project for this section (demo_wp5_retain.zip), please refer Chapter 12 to restore this project (Execute File> Add Existing Project > From Zip) and set the current IP address of your PAC.

This project includes an ST main program (Main) and 3 ST sub-programs (load_retain, save_retain and Go_retain).



Limitation :

This project is not good at handling Retain variables which value changes frequently. For example, value changed about every second or every minute. That is because these retain values of this project are saved within files in the \System_Disk. The file operations in it consume more CPU time, which will slow down the PAC performance if retain value changed frequently.

The "demo_wp5_retain" project has the following default values. Please set your own proper initial values.

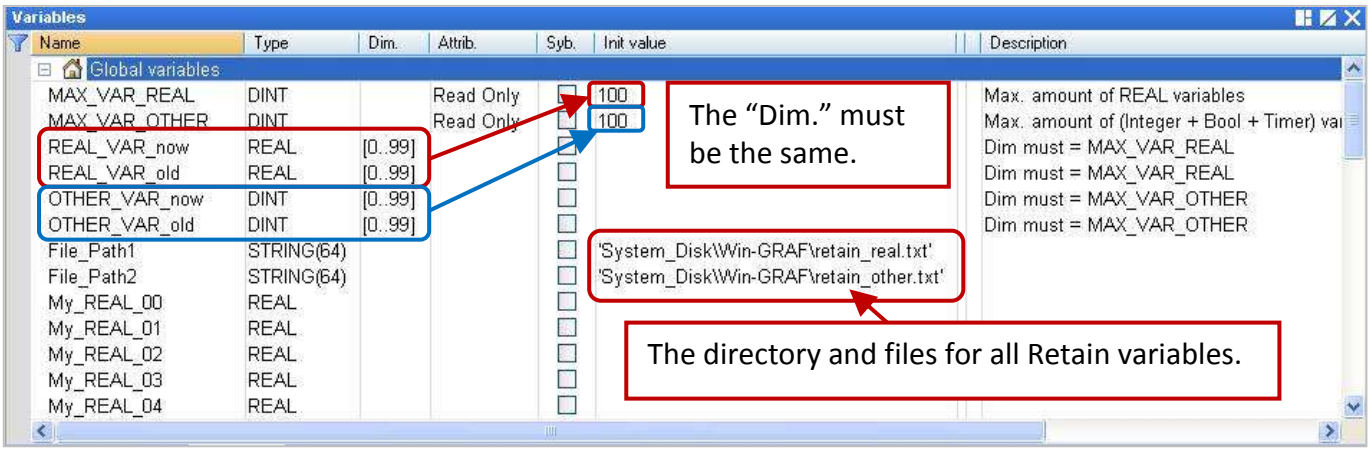
MAX_VAR_REAL: 100 (Max. amount for retaining REAL variables)
MAX_VAR_OTHER: 100 (Max. amount for retaining integer+Bool+Timer variables)
Dim of REAL_VAR_now and REAL_VAR_old: 100 ([0..99], the same as the value "MAX_VAR_REAL")
Dim of OTHER_VAR_now and OTHER_VAR_old: 100 ([0..99], the same as the value "MAX_VAR_OTHER")

Note: The bigger the value of the MAX_VAR_REAL and MAX_VAR_OTHER, the larger the PAC cycle time will be. So it is better to declare just fine value for the MAX_VAR_REAL and MAX_VAR_OTHER.

This project declares the following variables for demo purpose only. Please re-declare your own variables.

My_REAL_00 ~ My_REAL_07 (Data Type: REAL)
My_DINT_00 ~ My_DINT_05 (Data Type: DINT)
My_BOOL_00 ~ My_BOOL_02 (Data Type: BOOL)
My_Timer_00 ~ My_Timer_05 (Data Type: TIME)

You can see more variables in the "Variables" window.



"Go_retain" sub-program is used to do the retain operation. Remember to modify this sub-program. There are 4 sections should to be modified in it. Please search "Add your REAL variables for retain here" and "Add your integer, BOOL, Timer variables for retain here", and depend on your re-declared variables to modify your "Go_retain" sub-program.

```
(* Add your REAL variables for retain here *)
(* ---------------------------------------------- *)
My_REAL_00 := REAL_VAR_now[0] ;
My_REAL_01 := REAL_VAR_now[1] ;
My_REAL_02 := REAL_VAR_now[2] ;
My_REAL_03 := REAL_VAR_now[3] ;
My_REAL_04 := REAL_VAR_now[4] ;
My_REAL_05 := REAL_VAR_now[5] ;
My_REAL_06 := REAL_VAR_now[6] ;
My_REAL_07 := REAL_VAR_now[7] ;
(* ---------------------------------------------- *)
```

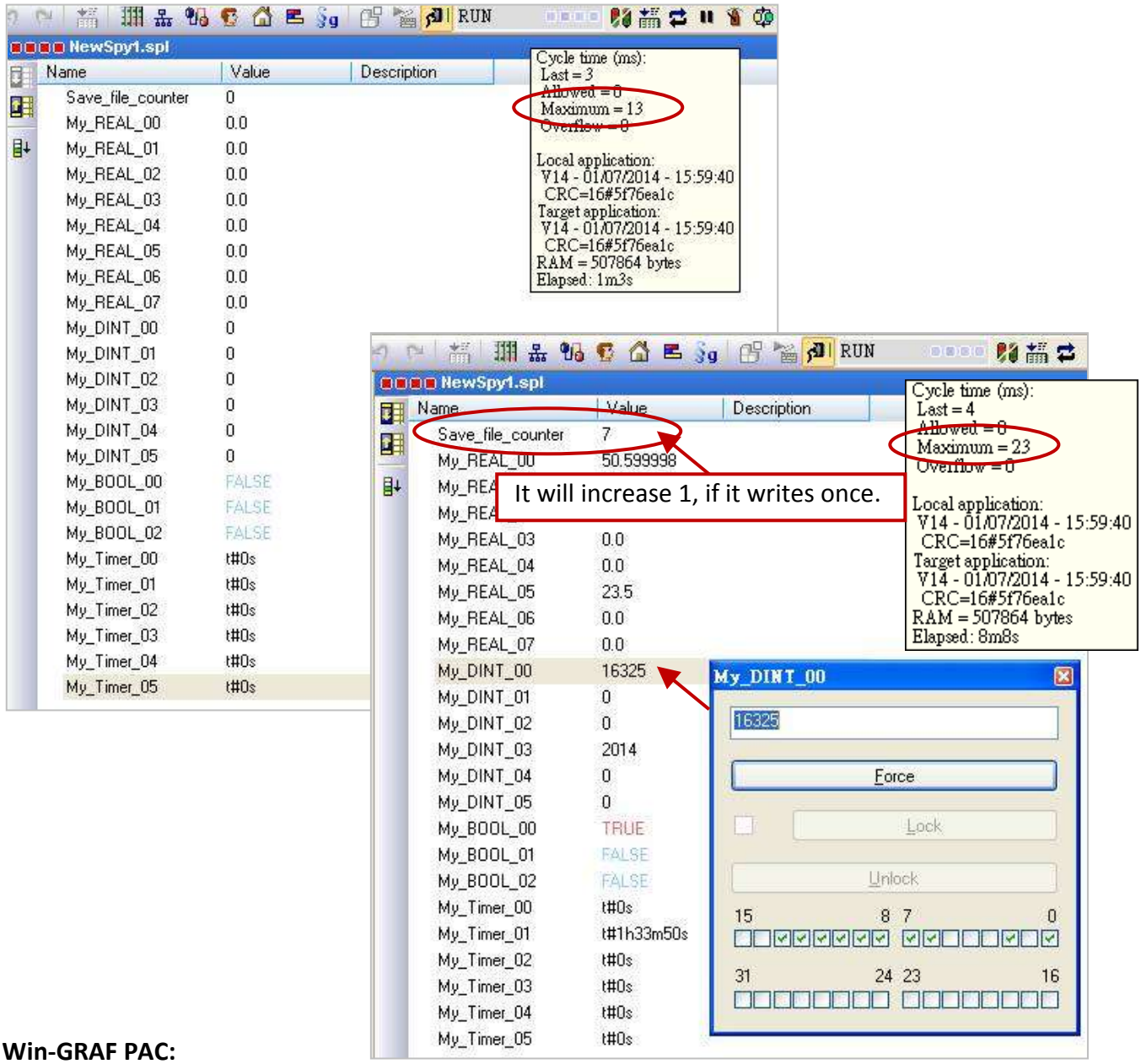
```
(* Add your integer, BOOL, Timer variables for retain here *)
(* ............................................ *)
My_DINT_00 := OTHER_VAR_now[0] ;
My_DINT_01 := OTHER_VAR_now[1] ;
My_DINT_02 := OTHER_VAR_now[2] ;
My_DINT_03 := OTHER_VAR_now[3] ;
My_DINT_04 := OTHER_VAR_now[4] ;
My_DINT_05 := OTHER_VAR_now[5] ;
```

```
    My_BOOL_00 := Any_to_BOOL( OTHER_VAR_now[6] ) ;
    My_BOOL_01 := Any_to_BOOL( OTHER_VAR_now[7] ) ;
    My_BOOL_02 := Any_to_BOOL( OTHER_VAR_now[8] ) ;

    My_Timer_00 := Any_to_TIME( OTHER_VAR_now[9] ) ;
    My_Timer_01 := Any_to_TIME( OTHER_VAR_now[10] ) ;
    My_Timer_02 := Any_to_TIME( OTHER_VAR_now[11] ) ;
    My_Timer_03 := Any_to_TIME( OTHER_VAR_now[12] ) ;
    My_Timer_04 := Any_to_TIME( OTHER_VAR_now[13] ) ;
    My_Timer_05 := Any_to_TIME( OTHER_VAR_now[14] ) ;
    (* ............................................ *)


(* Add your REAL variables for retain here *)
 (* ------------------------------------------------ *)
 REAL_VAR_now[0]  := My_REAL_00 ;
 REAL_VAR_now[1]  := My_REAL_01 ;
 REAL_VAR_now[2]  := My_REAL_02 ;
 REAL_VAR_now[3]  := My_REAL_03 ;
 REAL_VAR_now[4]  := My_REAL_04 ;
 REAL_VAR_now[5]  := My_REAL_05 ;
 REAL_VAR_now[6]  := My_REAL_06 ;
 REAL_VAR_now[7]  := My_REAL_07 ;
 (* ------------------------------------------------ *)


 (* Add your integer, BOOL, Timer variables for retain here *)
 (* ............................................ *)
 OTHER_VAR_now[0]  := My_DINT_00 ;
 OTHER_VAR_now[1]  := My_DINT_01 ;
 OTHER_VAR_now[2]  := My_DINT_02 ;
 OTHER_VAR_now[3]  := My_DINT_03 ;
 OTHER_VAR_now[4]  := My_DINT_04 ;
 OTHER_VAR_now[5]  := My_DINT_05 ;

 OTHER_VAR_now[6]  := Any_to_DINT( My_BOOL_00 ) ;
 OTHER_VAR_now[7]  := Any_to_DINT( My_BOOL_01 ) ;
 OTHER_VAR_now[8]  := Any_to_DINT( My_BOOL_02 ) ;

 OTHER_VAR_now[9]  := Any_to_DINT( My_Timer_00 ) ;
 OTHER_VAR_now[10] := Any_to_DINT( My_Timer_01 ) ;
 OTHER_VAR_now[11] := Any_to_DINT( My_Timer_02 ) ;
 OTHER_VAR_now[12] := Any_to_DINT( My_Timer_03 ) ;
 OTHER_VAR_now[13] := Any_to_DINT( My_Timer_04 ) ;
 OTHER_VAR_now[14] := Any_to_DINT( My_Timer_05 ) ;
 (* ............................................ *)
```

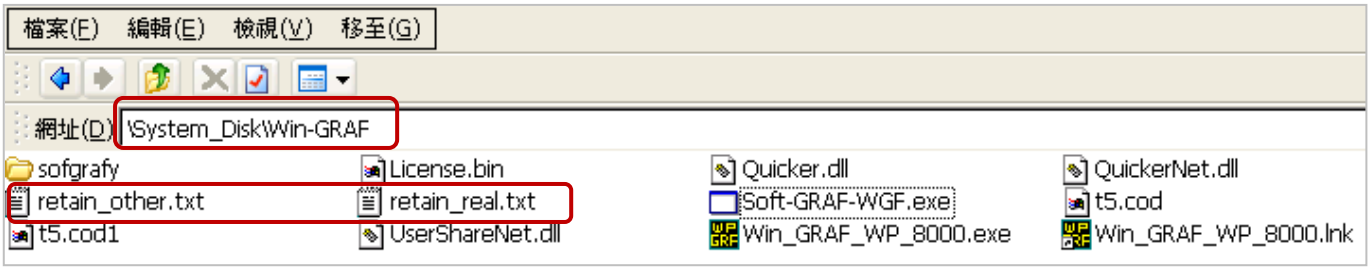
Test Project:

Before testing, make sure you have already set up the PAC IP and compile/download the project into the PAC (refer Section 2.3.4 and Section 2.3.5). When connecting with the PAC, all values in the Spy list (refer Section 11.3) will be "0" (or "FALSE") at the begging, please enter some values at will. When the value has changed, it will create a text file on the PAC (\System_disk\Win-GRAF\retain_real.txt and retain_other.txt) and Write data to the files.

Note: The "Save_file_counter" will show the number of times the file is written, if this value is changing rapidly (e.g., to write several times every second/minute.), it is not suitable for this application (Because to write into files frequently in the "\System_disk" will reduce the PAC effectiveness.).



Win-GRAF PAC:



6.3 Save Data to EEPROM

The Win-GRAF PAC has a built-in EEPROM memory for users to read and write data, which will not lose data when the PAC shut down. Compared to the read and write of the SRAM, EEPROM has the following disadvantages:

Note: Some PAC have no EEPROM memory (like the WP-5238-CE7). They don't support EEP_Read() and EEP_Write().

Advantages: Provides another way to save the important data, besides the Retain Variable. (Section 6.1).

Disadvantages: 1. The operation to read/write EEPROM will use much more CPU time (about 5 ~ 50 ms), but changing to the way of "Retain variable", CPU time is much less than 1 ms. Therefore, do not use the "EEP_Read" and "EEP_Write" Functions too frequently, or it will increase the PAC Cycle time.

2. EEPROM has a "write" limitation (depending on the PAC), it is not suitable to write the same data many times. So, **DO NOT** call "EEP_Write" Function within each PAC Cycle to do the "write" operation.

ST Program: (Following will show the safe and dangerous coding ways.**)**

```
(* Declare "FIRST_CYCLE" as a "BOOL" variable and has an initial value "TRUE".
   Declare "tmp_bool" as a "BOOL" variable。
   Declare "New_Val" and "Old_Val" as "DINT" variables. *)
(* Read the EEPROM once in the first Cycle. *)
if  FIRST_CYCLE  then
  FIRST_CYCLE  :=  FALSE ;  (* means it is not the first Cycle any more *)
  tmp_bool :=  EEP_Read ( 1 , New_Val) ;
end_if ;
(* Safe Coding Way: write to the EEPROM only when the value is changed.  *)
if  New_Val <> Old_Val  then
  Old_Val  :=  New_Val ;
  tmp_bool :=  EEP_Write ( 1 , New_Val ) ;
end_if ;
```

```
(* Dangerous Coding Way: EEPROM may be destroyed very soon. *)
(* Declare "FIRST_CYCLE" as "BOOL" variable and has an initial value "TRUE".
   Declare "tmp_bool" as "BOOL" variable.
   Declare "New_Val" and "Old_Val" as "DINT" variables. *)
(* Read the EEPROM once in the first Cycle. *)
if  FIRST_CYCLE  then
  FIRST_CYCLE  :=  FALSE ;  (* means it is not the first Cycle any more *)
  tmp_bool  :=  EEP_Read ( 1 , New_Val) ;
end_if ;
(*Dangerous Coding Way: Write the "New_Val" value to the EEPROM one time in every Cycle. *)
tmp_bool  :=  EEP_Write ( 1 , New_Val ) ;
```

6.3.1 EEP_READ (Read a Value from the EEPROM)

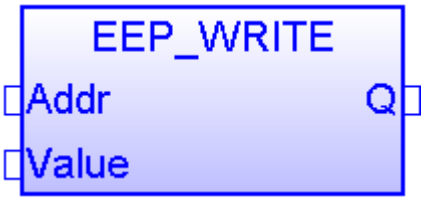
Tip:
Press “F1” key to see the detailed setting descriptions.

Addr: (Data Type: “DINT”)
Address, can be 1 to 1200. If the variable type of the "@Name" parameter is a 64-bit data (e.g., LINT or LREAL), the "Addr" can be 1001 to 1200 only.

@Name :
A variable name to store the value from the EEPROM.
(DO NOT use string variable. Variable type can be BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, LINT or LREAL.)

Q:
Data Type: BOOL. TRUE: Ok; FALSE: Error.
If the type of the "@Name" parameter is REAL or LREAL, will return "Q" as FALSE if the value is NaN (Not a Number) or other error happens. In the case of NaN, the REAL/LREAL variable will get the value "0.0".

6.3.2 EEP_WRITE (Write a Value to the EEPROM)



Addr: (Data Type: “DINT”)
Address, can be 1 to 1200. If the variable type of the "Value" parameter is a 64-bit data (e.g., LINT or LREAL), the "Addr" can be 1001 to 1200 only.

Value :
The value to write to the EEPROM.
(DO NOT use string variable. The value type can be BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, LINT or LREAL.)

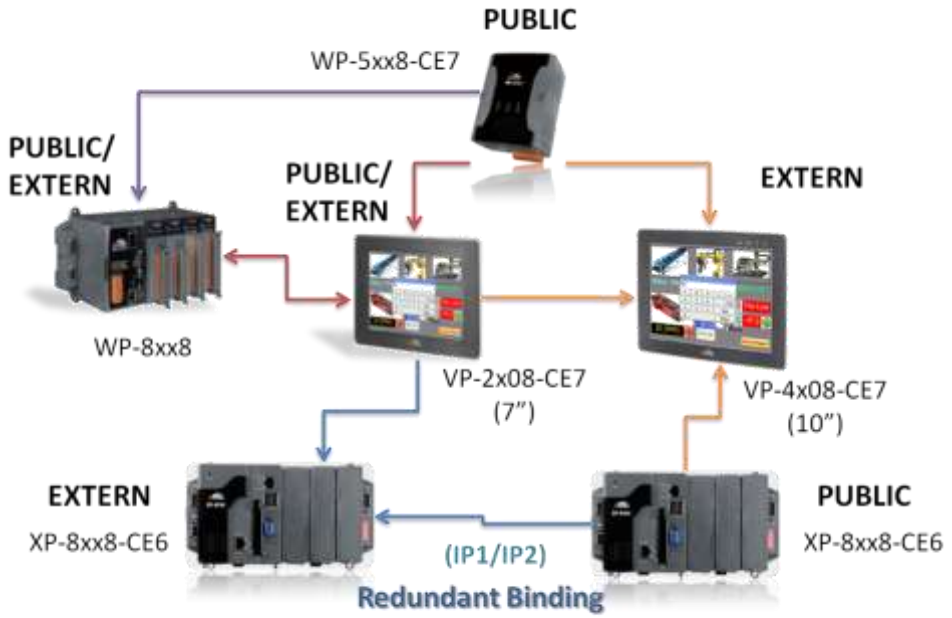
Q:
Data Type: BOOL. TRUE: Ok; FALSE: Error.

Chapter 7 Exchange Data between PACs (Data Binding)

"Binding" function is used to exchange data between ICP DAS Win-GRAF PACs, the data transmission is event triggered. It is much more efficient than polling way. Win-GRAF offers two ways to set up Binding:

- **PUBLIC:** Publish one PAC's own data, or for use of VB .net, C#, or C in the same PAC.
- **EXTERN:** To get data from other PAC.

Application Diagram:



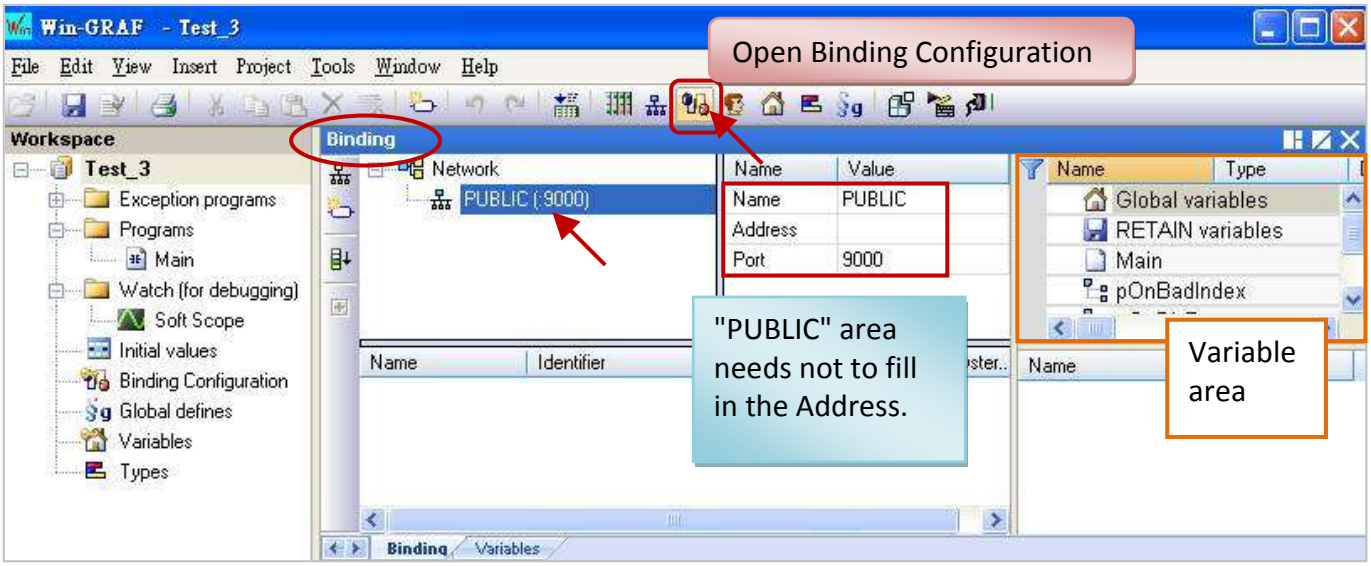
Note:
The max. number of "Binding" (EXTRN) can be used by one Win-GRAF PAC:
XPAC: Max. 32
WinPAC: Max. 16
ViewPAC: Max. 16
(See P1-1 for details.)

"PUBLIC" Setting Steps:

When the PAC sets up the "PUBLIC" area, means to publish its own data.

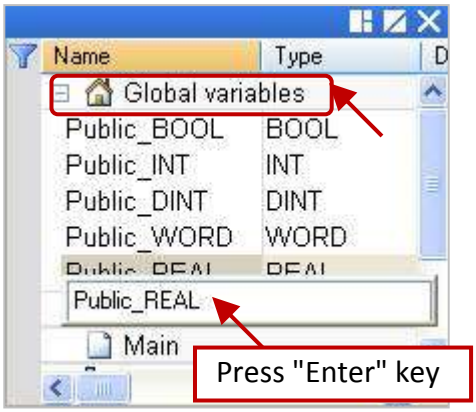
1. Mouse right-click on the toolbar "Open Binding Configuration" icon to open the "Binding" window.

2. Click "PUBLIC (: 9000)" to configure the data to be published as PUBLIC; "Address" field needs not to fill in; "Port" field is fixed to "9000", do not change it.

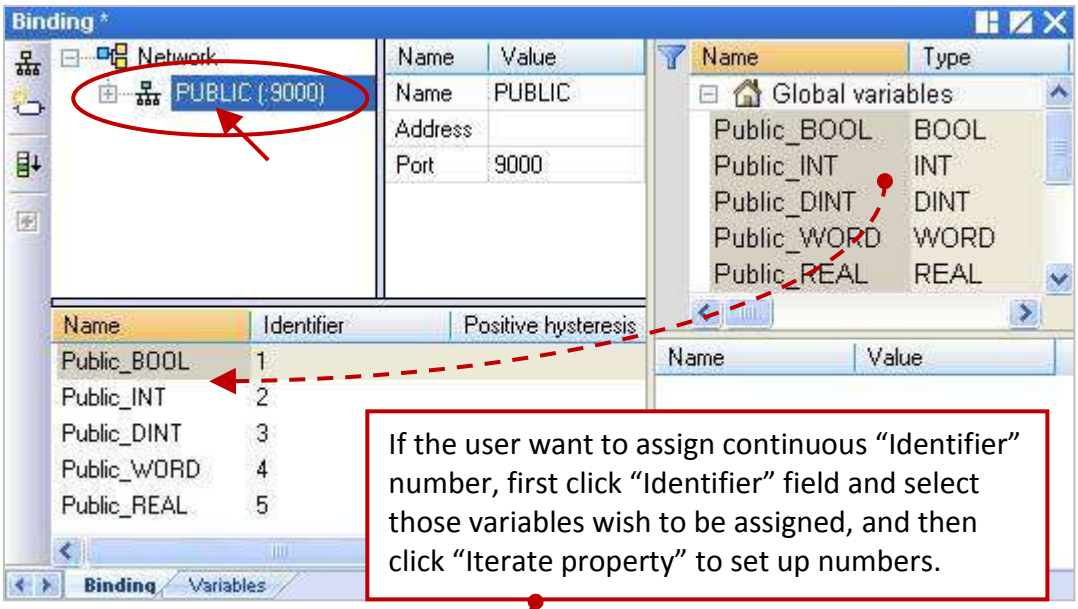


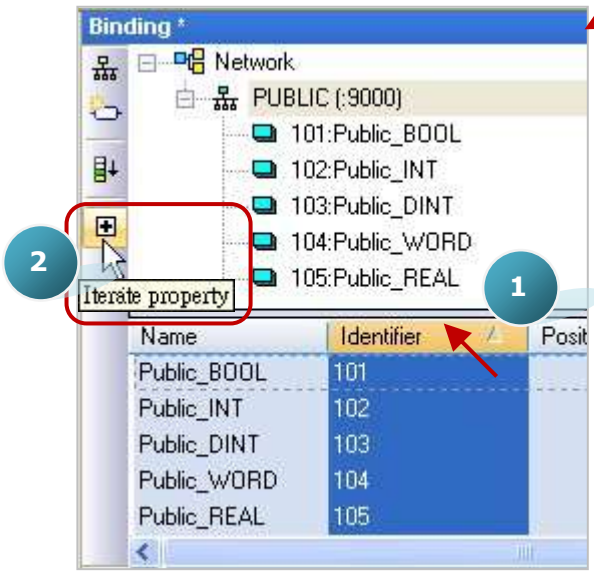
3. Before setting the data to become public, first you must create the variables want to become public in the Variable area. Right-click on "Global variables" and press "Ins" key to add the new variables. In the following table, the variables are used for this example. You can set up your own. After finished, the screen is as below.

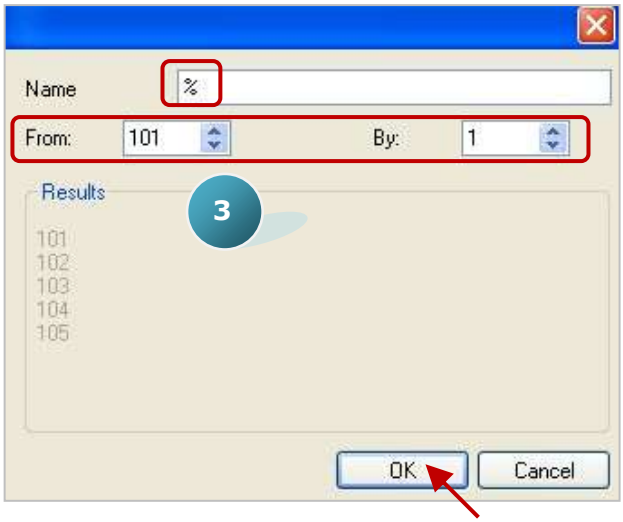
Variables Name	Data Type
Public_BOOL	BOOL
Public_INT	INT
Public_DINT	DINT
Public_WORD	WORD
Public_REAL	REAL



4. Right-click on the "PUBLIC (: 9000)", and then select the variables that you want to publish, and drag them to the "Name" area. "Identifier" field will automatically generate numbers (if other PAC wants to access the data, need to set the same ID number).
Note: "PUBLIC" can use up to 8192 variables; "Identifier" number can only be "1 ~ 8192".



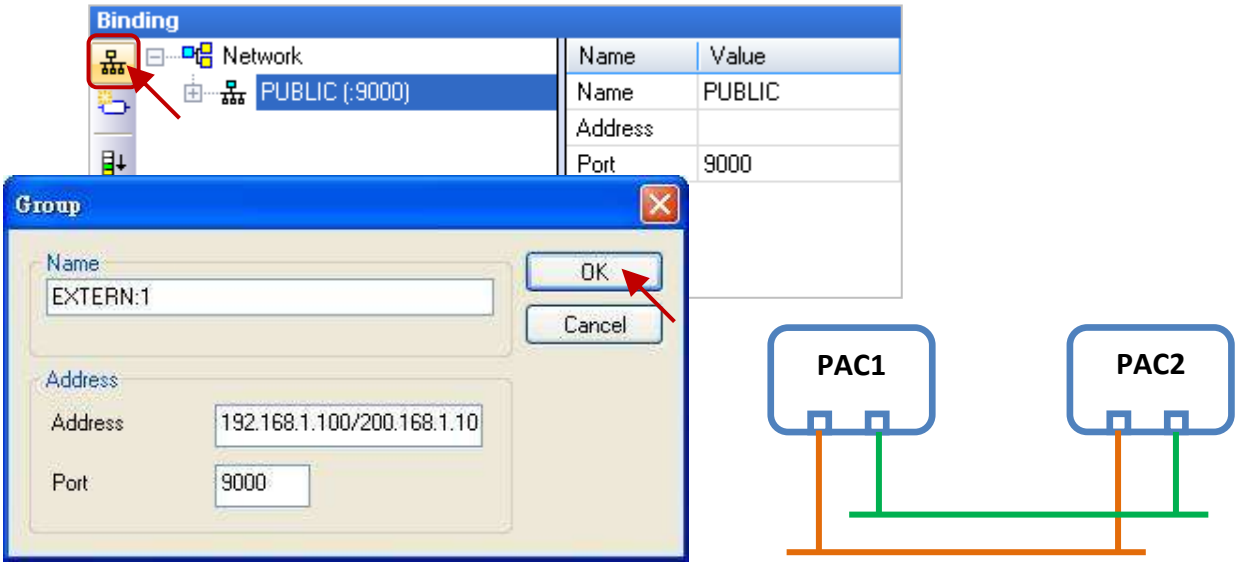




"EXTERN" Setting Steps:

When the PAC sets up the "EXTERN" area, means will get the data from other external PAC.

5. Click the "Insert Master/Port" icon in the left side, it will show the "Group" window. Follow the description below to set up this window, and then click "OK" button.



Name: Can be modified to the desired name.

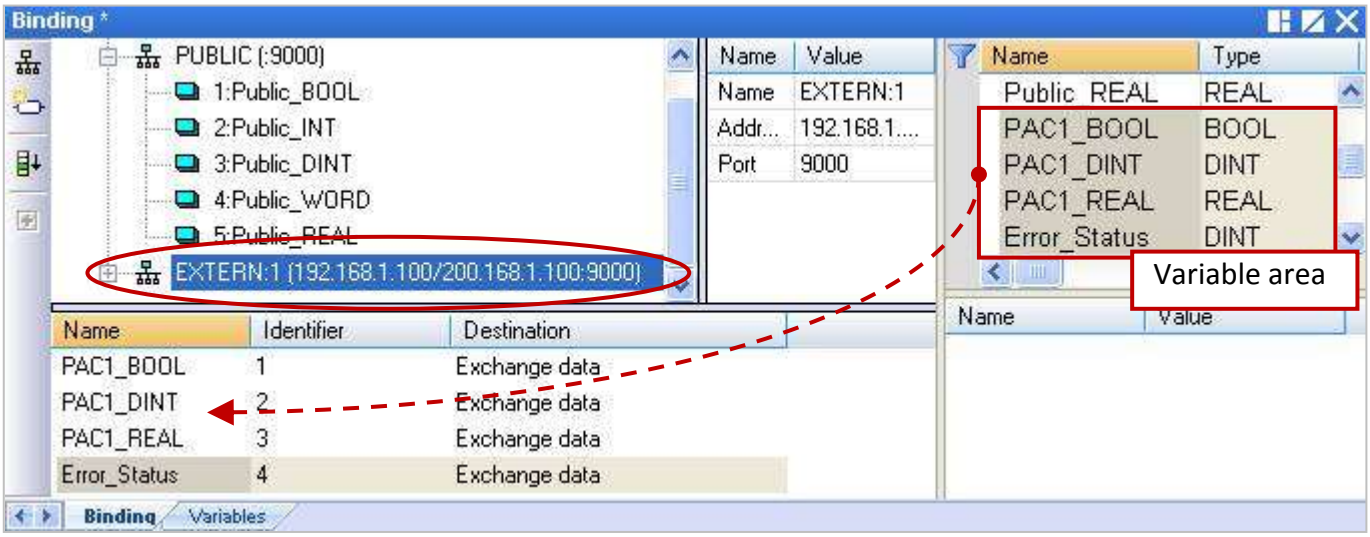
Address: Enter the IP address of the PAC which data will be obtained (e.g., "192.168.1.100"). User can enter two IP Addresses (e.g., "192.168.1.100/200.168.1.100"; that PAC must use two Ethernet Ports), so that when one IP address occurs any problem, it will try to link the second IP address.

Port: Fixed to use "9000", do not change it.

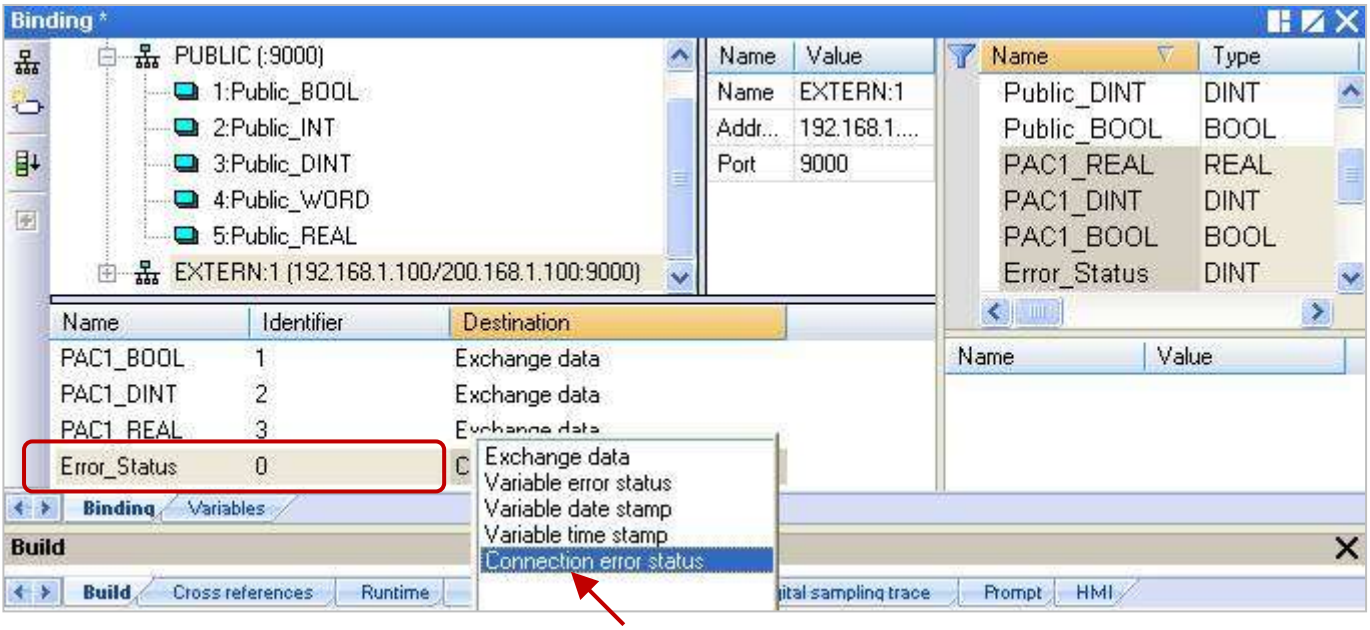
6. In the variable area, set up the data type you want to get. (Refer Step 3 - Right click on "Global variables" and press "Ins" key to add the variables). The variables that listed in the table are used for this example. You can set up your own. After finished, the screen is as below.

Variables Name	Data Type
PAC1_BOOL	BOOL
PAC1_DINT	DINT
PAC1_REAL	REAL
Error_Status	DINT

7. Please drag the variables you need into the "Name" area of the "EXTERN:1".
Note: "Identifier" field will automatically generate numbers, please change them to the same as the opened IDs of the PAC that you want to get data from.



8. As the picture below, "Error_Status" variable is used to determine the communication status of the PAC, please set this ID to "0" and then double-click "Destination" field and set it to "Connection error status".



Note:

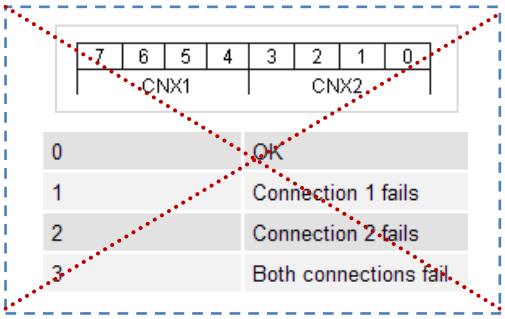
a. If "EXTERN" set up two IP addresses (step 5), then the "Error_Status" will return two communication status. As the following table, use 8 bits to indicate the connection status. The bit 0 ~ 3 represents the first IP connection status (If all bits are 1, the value is 15); the bit 4 ~ 7 represents the second IP connection status (If all bits are 1, the value is 240). As long as the value is not equal to "0", it means that there are connection errors.

IP2 Connection Status				**IP1** Connection Status				**Status Description**
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
0				0				Connection OK
0				≠0 (1 ~ 15)				IP1 Connection error
≠ 0 (16 ~ 240)				0				IP2 Connection error
≠ 0				≠ 0				IP1 and IP2 Connection error

b. The return value of the "Error_Status" is an integer value. The following division operation provides a way to determine this value. Dividing this value by 16, the quotient represents the IP2 connection status, and the remainder represents the IP1 connection status. If the values are not equal to "0", it means that there is any connection error. For example: If "Error_Status" = 16, divided by 16, the quotient = 1 (≠ 0, IP2 Connection error) and the remainder = 0 (IP1 Connection OK); If "Error_Status" = 3, divided by 16, the quotient = 0 (IP2 Connection OK) and the remainder = 3 (≠ 0, IP1 Connection error);

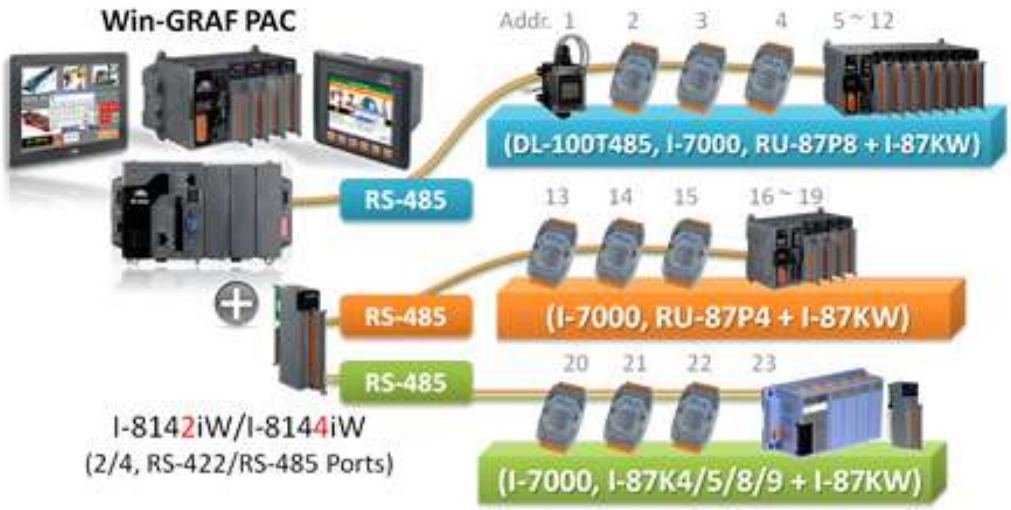
Notice:

Please ignore the description of the "Connection status" and "Variable status" in the "Networked applications - Dual binding on redundant ETHERNET" of the "HTML Help" that opened when press "F1" key. That explanation is wrong and does not comply with the ICP DAS Win-GRAF PAC. Please ignore it.



Chapter 8 Connecting DCON I/O Modules

The Win-GRAF PAC can connect the ICP DAS "I-7000" and "I-87KW" remote DCON I/O modules via the COM Port (RS-485). Each PAC can enable up to 16 DCON Ports, and each Port can connect up to 50 remote DCON I/O modules (not recommended over 32). If select the "I-87KW" series I/O modules, it must be used with the RS-485 I/O Expansion Unit (e.g., I-87K4/5/8/9 or RU-87P4/8). You can view the detailed product information on the ICP DAS website:
http://www.icpdas.com/root/product/solutions/remote_io/remote_io_products.html



Before connecting "I-7000" or "I-87KW" remote DCON I/O modules, you must use "DCON Utility" software to configure each module for the Protocol (choose DCON mode), Address (1 ~ 255), Baudrate (the setting must the same with the Win-GRAF PAC, recommended set to 9600), Checksum (the setting must the same with the Win-GRAF PAC, recommended set to "enabled" for communications security), Data format and other Input/Output settings (set according to demand).

Note:

A. When using the AI module of I-7000 and I-87KW, set the Data format to **"2's Complement"**.
   E.g. I-7005, I-7013, I-7014D, I-7015, I-7016, I-7017R, I-7018Z, I-7019R, I-7033; I-87005W, I-87013W, I-87015W, I-87015PW, I-87016W, I-87017W, I-87017RCW, I-87017ZW, I-87017DW, I-87018W, I-87018RW, I-87018ZW, I-87019RW, I-87019ZW, and other Analog Input modules.

B. When using the AO module of I-7000 and I-87KW, set the Data format to **"Engineering"**
   E.g. I-7021, I-7022, I-7024, I-7024R; I-87024W, I-87024UW, I-87024CW, I-87028UW, I-87028CW, I-87028VW, I-87028VW-20V, and other Analog Output modules.

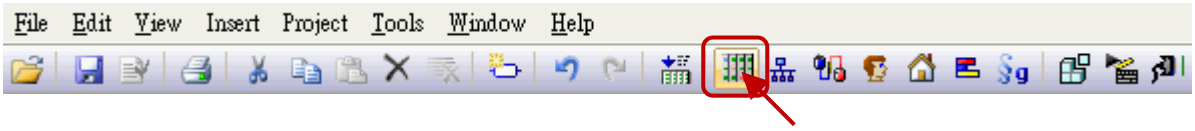
"DCON Utility" is an easy-to-use software toolkit that help user search the network, configure the I/O modules and test the I/O status. Please visit the website to get the software program and user manual:
www.icpdas.com/products/dcon/introduction.htm

The following will introduce the setting method in the Win-GRAF Workbench.

8.1 Setting "DCON" I/O Boards

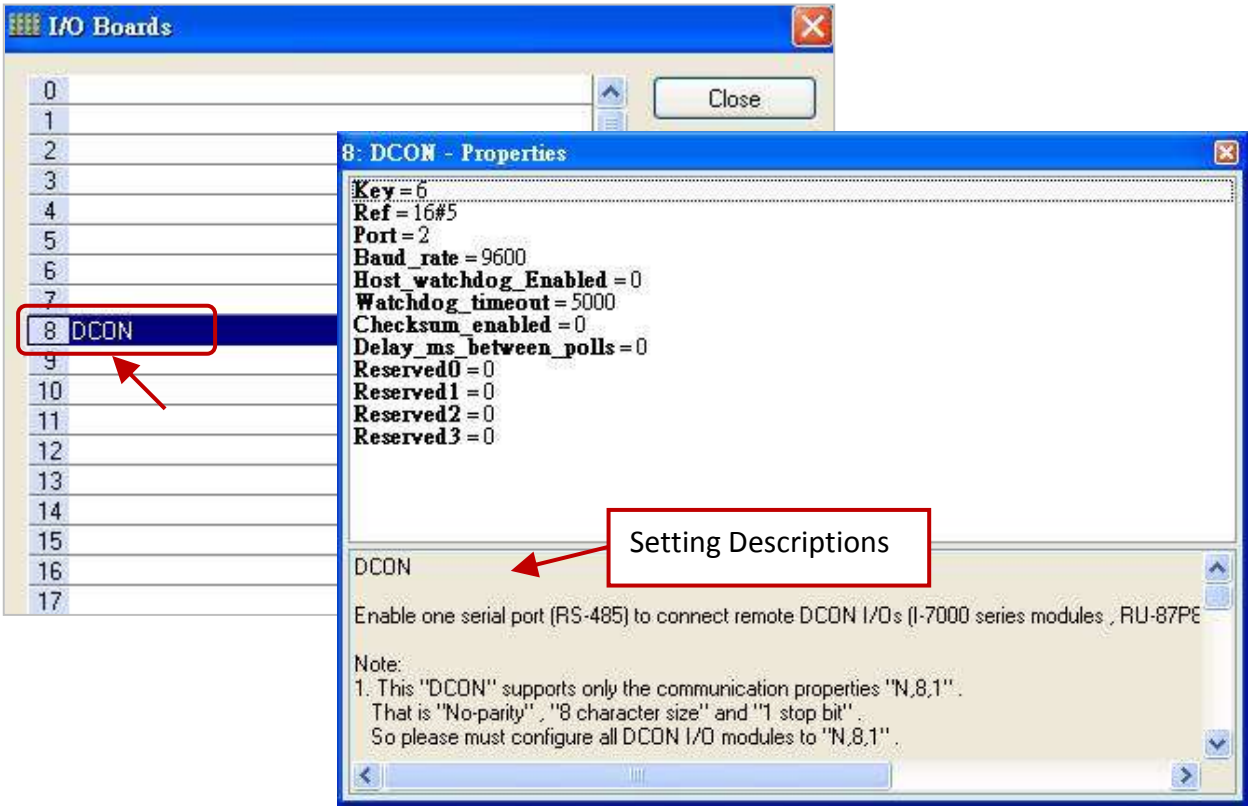
"DCON" can be used to enable an RS-485 Port to connect remote DCON I/O modules (e.g., I-7000 series modules, RU-87P8 I/O Expansion Unit + I-87KW I/O modules, or I-87K8 I/O Expansion Unit + I-87KW I/O modules). If want to enable more than one DCON Port, please set up multiple "DCON" I/O Boards. (One PAC can enable up to 16 "DCON".)

1. Click "Open I/Os" of the Win-GRAF tool bar to open the “I/O Boards” setting window.



2. Double click "Slot8" to add "DCON" I/O Boards (Refer Chapter 4), and then double click "DCON" to open the “Properties” window.

   Note: The Slot 0 to Slot 7 are reserved for real I/O modules that plugged into the PAC, and the slot 8 or above are for other usage.



Parameters:

Note: This "DCON" supports only the communication properties "N,8,1". That is "No-parity" , "8 character size" and "1 stop bit" . So please must configure all DCON I/O modules to "N,8,1".

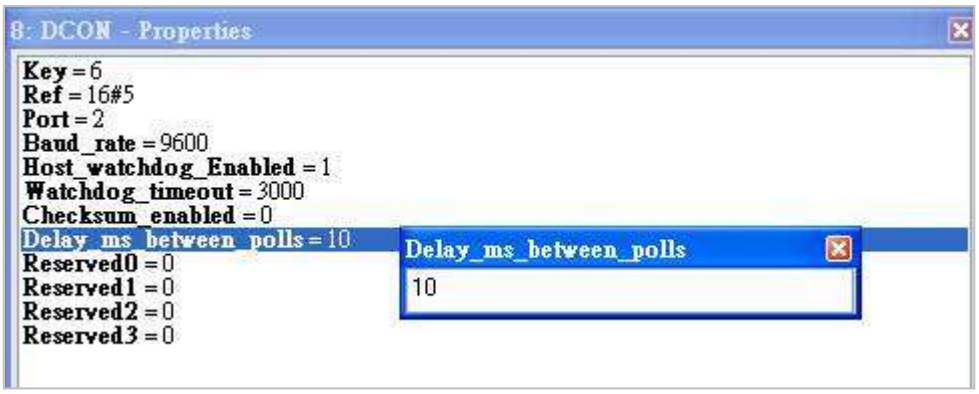
Port:	COM port number (1 ~ 37, depends on the PAC.)
Baud_rate:	Communication baudrate in bps, can be 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 (bps). Set a wrong value will use the default value 9600.
Host_watchdog_Enabled:	1: enable host-watchdog, 0: disable it. Set a nonzero value will use the value 1.

Watchdog_timeout: Unit: ms, can be 3000 ~ 25500.
Set larger than 25500 will use 25500 ms (25.5 sec).
Set smaller than 3000 will use 3000 ms (3 sec).
Ignore this setting when "Host_watchdog_Enabled" is 0.

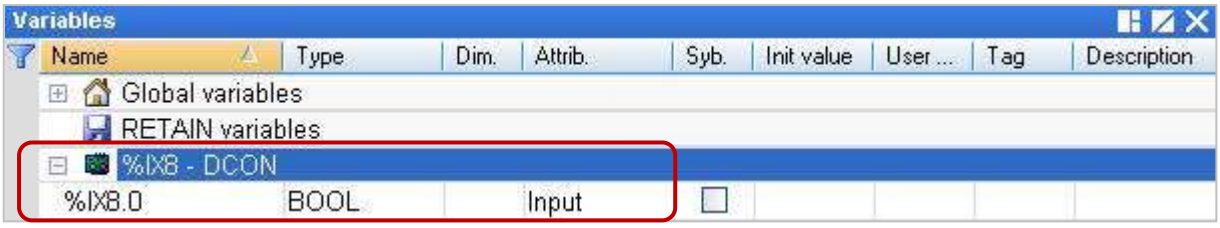
Checksum_enabled: 0: disabled, 1: enabled.
Set a nonzero value will use the value 1. (Recommended set to "enabled" for communications security.)

Delay_ms_between_polls: Unit: ms, default is 0 ms. Valid range is 0 ~ 1000.
Set smaller than 0 will use 0 ms.
Set larger than 1000 will use 1000 ms.
If there is no wireless module connected, set a smaller value.
For instance, set as 0 ~ 10.
If there are wireless modules (e.g., ICP DAS ZigBee Products: ZigBee Converters or ZigBee I/O modules.) connected, set a bigger value.
For instance, set a value between 30 ~ 100 or other values.
Set larger value will get slower polling efficiency.

Double click the item to be set, and then fill in the value.

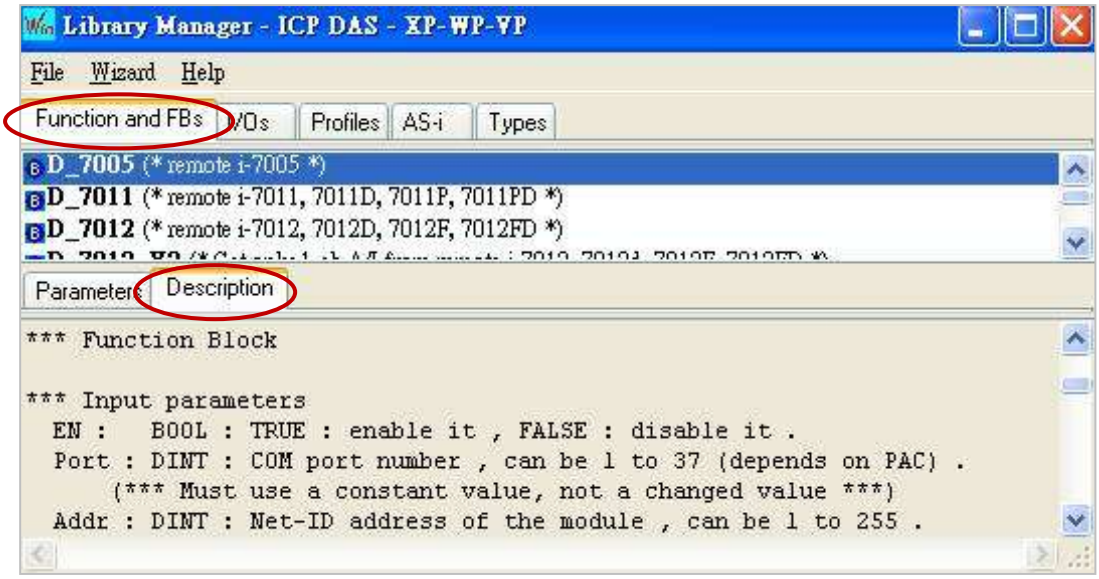


3. After setting up the "DCON" in the "I/O Boards" window, it will automatically add a "BOOL" input variable in the "Variables" window. When the Win-GRAF links to the PAC, it will show the COM Port communication status (TRUE: OK; FALSE: error.).

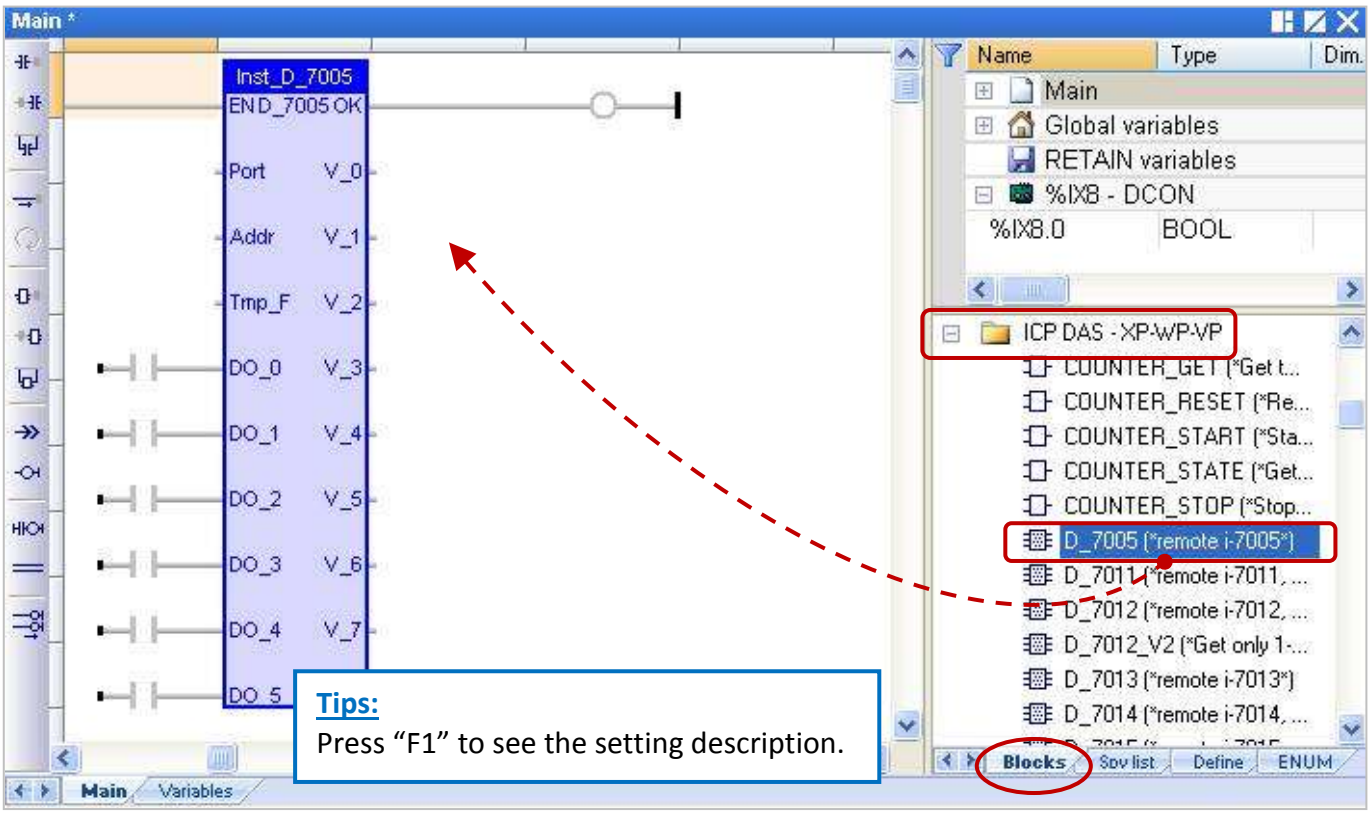


8.2 Using I/O Function Blocks

The Win-GRAF supports many ICP DAS DCON remote I/O modules, you can open the "Library Manager" (Refer Section 1.2.3) or press "F1" key in the "Function and FBs" to view these I/O Function Block descriptions. This section will introduce "D_7065", "D_7018Z", "D_7083", "D_87084_freq", "D_87084_cnt4", "D_87084_cnt8", and "DL_100T485" I/O Function Blocks.



In the LD Program - Functional Block area, expand the "ICP DAS - XP-WP-VP" folder of the "Blocks" panel. There are many Functions and Function Blocks. You can select the desired one, and drag it into the editing area of the program.



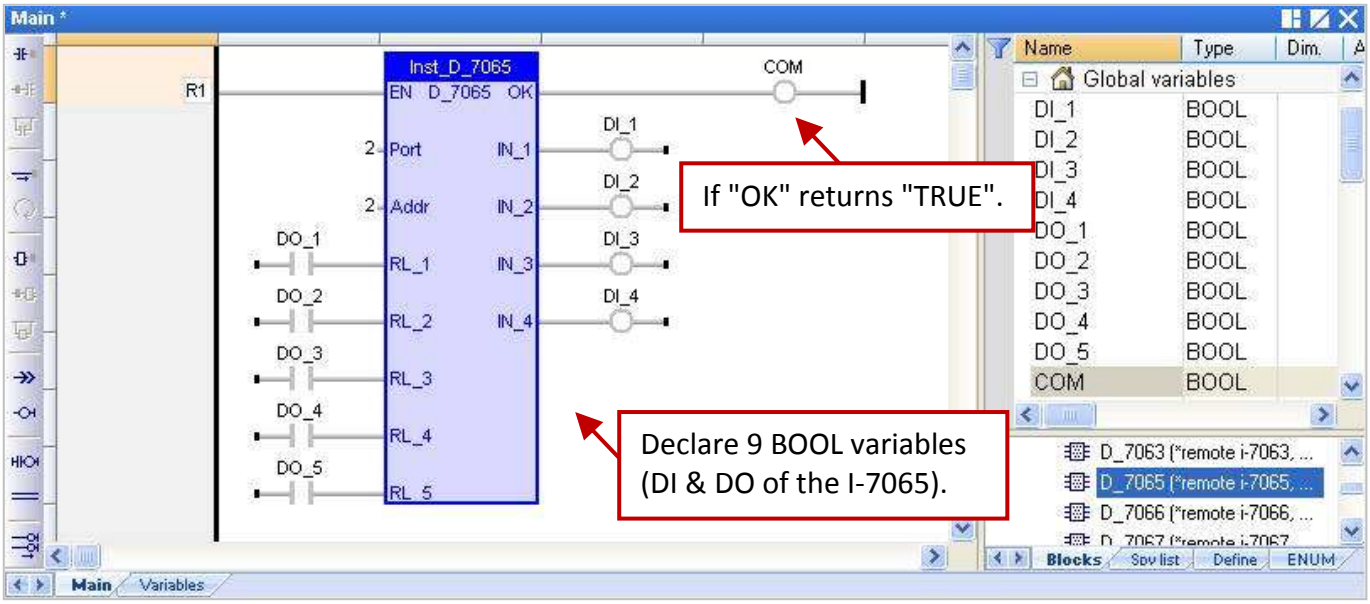
8.2.1 "D_7065" Function Block

"D_7065": Connect a remote I-7065, I-7065D (Power Relay Output Module) or I-7065A, I-7065AD, I-7065B, I-7065BD (Solid State Relay Output Module).

Note:

1. All connected DCON I/O modules should be configured once by the DCON Utility (see P8-1).
2. Please use "DCON"(Section 8.1) in the "I/O boards" window and set proper settings (Port, baud_rate, etc.) on it.
3. All values of DI channels are meaningful only when the returned communication state is TRUE (If "OK" returns "TRUE".).
4. Referring Chapter 12, click the menu bar "File" > "Add Existing Project" > "From Zip" to restore the demo project (CD-ROM: \Napdos\Win-GRAF\demo-project\DEMO_D_7065.zip) in the shipping CD and see the program and descriptions.

Supposition: Use PAC's COM2 to connect the I-7065 (Addr. = 2) with 4 DI and 5 Relay output channels.



Input Parameters:

EN:	Data type: BOOL. TRUE: enable it; FALSE: disable it.
Port:	Data type: DINT. COM port number (can be 1 to 37, depends on PAC). (*** Must use a constant value, cannot be a changed value. ***)
Addr:	Data type: DINT. The Net-ID address of the module , can be 1 to 255. (*** Must use a constant value, not a changed value ***)
RL_1 ~ RL_5:	Data type: BOOL. 5-Ch DO values.

Output Parameters:

OK:	Data type: BOOL. TRUE: Communication is Ok. FALSE: Communication failed.
IN_1 ~ IN_4:	Data type: BOOL. 4-Ch DI values.

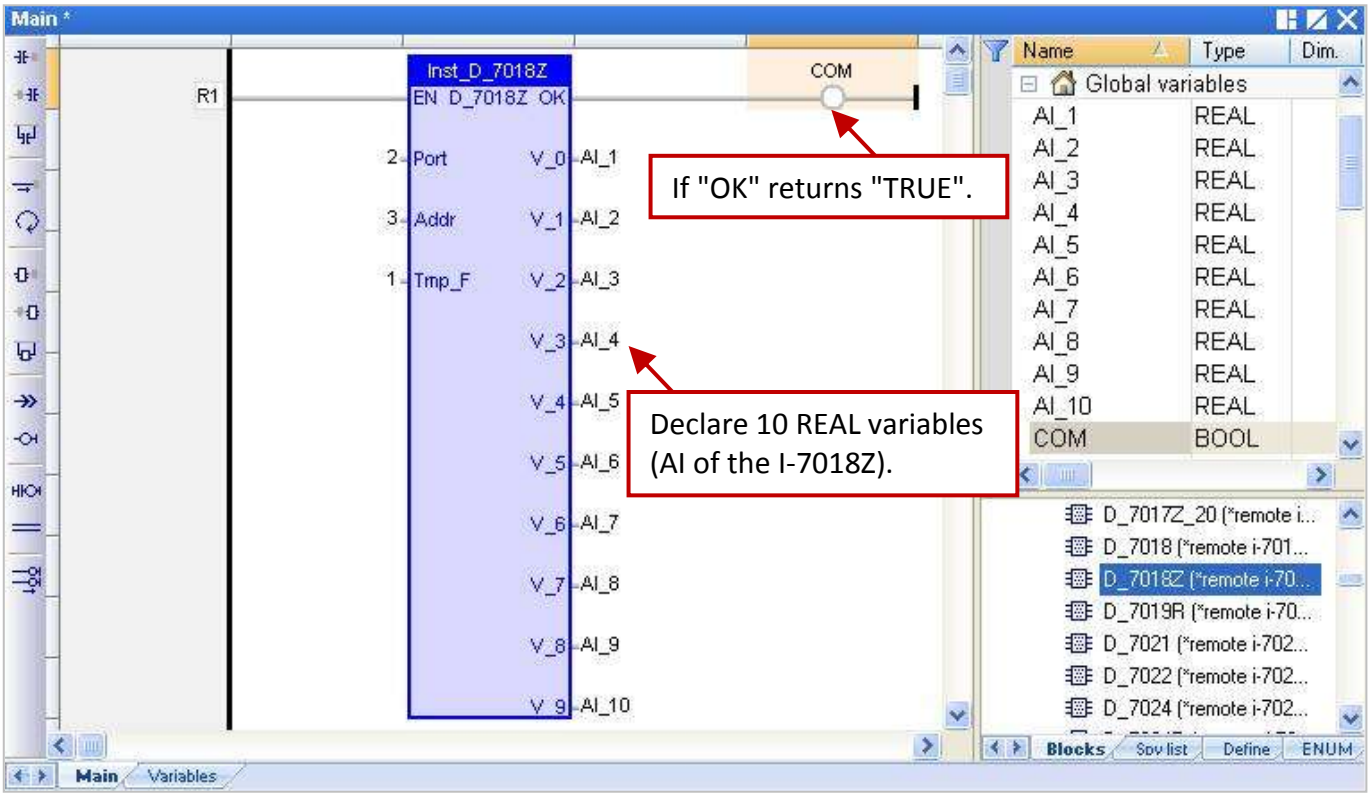
8.2.2 “D_7018Z” Function Block

“D_7018Z”: Connect a remote I-7018Z module that is a 10-channel Thermocouple analog input module for measuring voltage, current or temperature with features of individual channel configuration, open-wire detection and over Voltage protection.

Note:

1. All connected DCON I/O modules should be configured once (e.g., Address, Baudrate, etc.) by the DCON Utility (see P8-1). Please must configure the data format of AI modules to "**2’s complement**" by DCON utility, or the Win-GRAF PAC can not read them well.
2. Please use "DCON"(Section 8.1) in the "I/O boards" window and set proper settings (Port, baud_rate, etc.) on it.
3. All values of AI channels are meaningful only when the returned communication state is TRUE (If "OK" returns "TRUE".).
4. Referring Chapter 12, click the menu bar "File" > "Add Existing Project" > "From Zip" to restore the demo project (CD-ROM: \Napdos\Win-GRAF\demo-project\DEMO_D_7018z.zip) in the shipping CD and see the program and descriptions.

Supposition: Use PAC‘s COM2 to connect the I-7018Z (Addr. = 3) to measure the Celsius temperature.



Input Parameters:

EN:	Data type: BOOL. TRUE: enable it; FALSE: disable it.
Port:	Data type: DINT. COM port number (can be 1 to 37, depends on PAC). (*** Must use a constant value, cannot be a changed value. ***)
Addr:	Data type: DINT. The Net-ID address of the module , can be 1 to 255. (*** Must use a constant value, not a changed value ***)

Tmp_F: Data type: DINT. Temperature Format, can be 1 or 2:
1 : temperature unit in Degree Celsius.
2 : temperature unit in Degree Fahrenheit.
Other value: use it as "1:temperature unit in Degree Celsius".

Output Parameters:

OK: Data type: BOOL. TRUE: Communication is Ok. FALSE: Communication failed.

V_0 ~ V_9: Data type: REAL. 10-Ch AI value.
If the channel range type is configured as mV or Volt by DCON utility, the unit of the returned channel value is Volt.
For example, 0.85421 means 0.85421 V or 854.21 mV.
If the channel range type is configured as mA by DCON utility, the unit of the returned channel value is mA.
For example, 1.5567 means 1.5567 mA .
If the channel range type is configured as temperature, the value unit is degree.
For example, 25.75 means 25.75 degrees.

Open-wire Detection:

If the returned temperature is greater than "9000.0", it means that
1. The temperature sensor may be broken-line.
2. The temperature sensor may be damaged.
3. The DCON module is not configured well to fit the connected temperature sensors.
4. The ohm measured by the connected sensor is not correct.

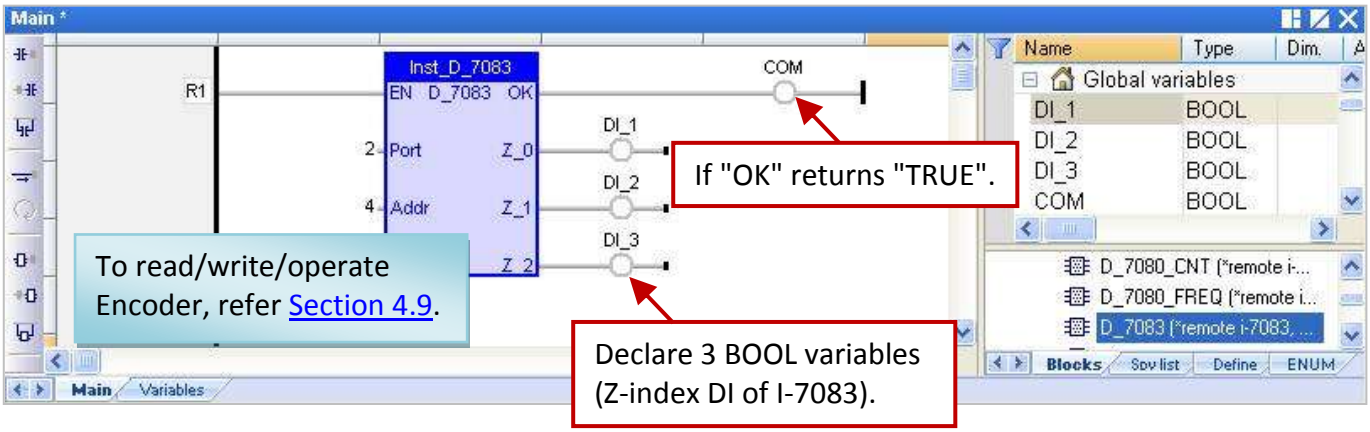
8.2.3 “D_7083” Function Block

“D_7083”: Connect a remote I-7083, I-7083D, I-7083B or I-7083BD module that is a 3-axis, 32 bits encoder counter.

Note:

1. To get the Encoder value of the I-7083, I-7083D, I-7083B and I-7083BD module, first using "D_7083" Function Block. Then, using the “Counter_Start”, “Counter_Stop”, “Counter_Get”, “Counter_State” and “Counter_Reset” Functions (Refer Section 4.9) to operate encoder channels in an I-7083, I-7083D, I-7083B and I-7083BD module.
2. All connected DCON I/O modules should be configured once (e.g., Address, Baudrate, etc.) by the DCON Utility (see P8-1). Please must configure the data format of AI modules to "**2’s complement**" by DCON utility, or the Win-GRAF PAC can not read them well.
3. Please use "DCON"(Section 8.1) in the "I/O boards" window and set proper settings (Port, baud_rate, etc.) on it.
4. All values of AI channels are meaningful only when the returned communication state is TRUE (If "OK" returns "TRUE".).
5. Referring Chapter 12, click the menu bar "File" > "Add Existing Project" > "From Zip" to restore the demo project (CD-ROM: \Napdos\Win-GRAF\demo-project\DEMO_D_7083.zip) in the shipping CD and see the program and descriptions.

Supposition: Use PAC’s COM2 to connect the I-7083 (Addr. = 4) with 3 DI channels.



Input Parameters:

EN:	Data type: BOOL. TRUE: enable it; FALSE: disable it.
Port:	Data type: DINT. COM port number (can be 1 to 37, depends on PAC). (*** Must use a constant value, cannot be a changed value. ***)
Addr:	Data type: DINT. The Net-ID address of the module , can be 1 to 255. (*** Must use a constant value, not a changed value ***)

Output Parameters:

OK:	Data type: BOOL. TRUE: Communication is Ok. FALSE: Communication failed.
Z_0 ~ Z_2:	Data type: BOOL. 3-ch Z-index DI value.

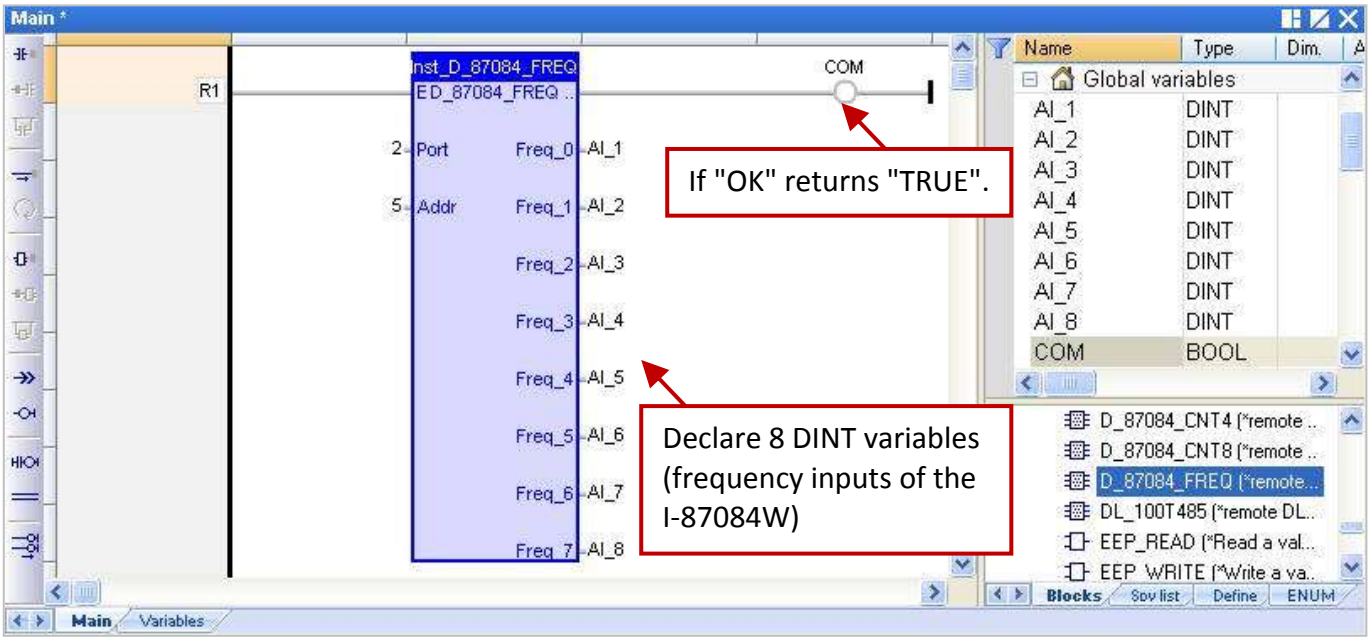
8.2.4 "D_87084_FREQ" Function Block

"D_87084_freq": Connect a remote I-87084W in an I/O Expansion Unit (e.g., I-87K4/5/8/9 or RU-87P4 or RU-87P8.) to measure 8-ch frequency.

Note:

1. Please MUST configure the I-87084W's frequency data format as "**Hex format**" by DCON utility (see P8-1) when using the I-87084W to measure frequency. Or it will not work.
2. Please use "DCON"(Section 8.1) in the "I/O boards" window and set proper settings (Port, baud_rate, etc.) on it.
3. All values of AI channels are meaningful only when the returned communication state is TRUE (If "OK" returns "TRUE".).
4. Referring Chapter 12, click the menu bar "File" > "Add Existing Project" > "From Zip" to restore the demo project (CD-ROM: \Napdos\Win-GRAF\demo-project\DEMO_D_87084_FR.zip) in the shipping CD and see the program and descriptions.

Supposition: Use PAC's COM2 to connect the I-87084W (Addr. = 5) to measure 8-ch frequency.



Input Parameters:

EN:	Data type: BOOL. TRUE: enable it; FALSE: disable i**t.**
Port:	Data type: DINT. COM port number (can be 1 to 37, depends on PAC). (*** Must use a constant value, cannot be a changed value. ***)
Addr:	Data type: DINT. The Net-ID address of the module , can be 1 to 255. (*** Must use a constant value, not a changed value ***)

Output Parameters:

OK:	Data type: BOOL. TRUE: Communication is Ok. FALSE: Communication failed.
Freq_0 ~ Freq_7:	Data type: DINT. 8-Ch frequency value, unit is Hz.

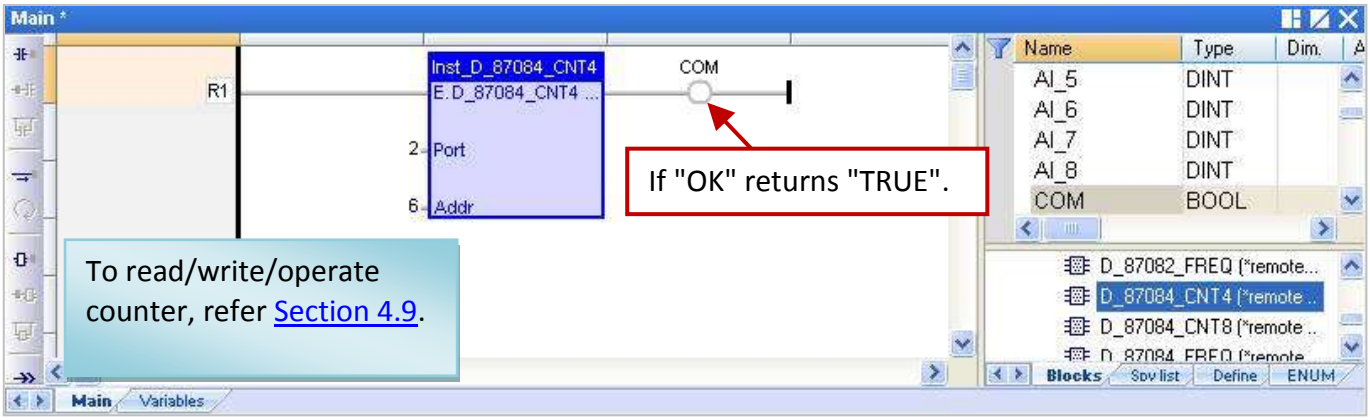
8.2.5 "D_87084_CNT4" Function Block

"D_87084_CNT4": Connect a remote I-87084W in an I/O Expansion Unit (e.g., I-87K4/5/8/9 or RU-87P4 or RU-87P8.) to measure 4-ch counters.

Note:

1. Please MUST configure the I-87084W's counter data format as "**Hex format**" by DCON utility (see P8-1) when using the I-87084W to measure counters. Or it will not work.
2. Please use "DCON"(Section 8.1) in the "I/O boards" window and set proper settings (Port, baud_rate, etc.) on it.
3. To get the 4-ch counter value from the remote I-87084W, first using "D_87084_cnt4" Function Block. Then, using the "Counter_Start", "Counter_Stop", "Counter_Get", "Counter_State" and "Counter_Reset" Functions (refer Section 4.9) to operate counter channels.
4. All values of AI channels are meaningful only when the returned communication state is TRUE (If "OK" returns "TRUE".).
5. Referring Chapter 12, click the menu bar "File" > "Add Existing Project" > "From Zip" to restore the demo project (CD-ROM: \Napdos\Win-GRAF\demo-project\DEMO_D_87084_C4.zip) in the shipping CD and see the program and descriptions.

Supposition: Use PAC's COM2 to connect the I-87084W (Addr. = 6) to measure 4-ch counters.



Input Parameters:

EN:	Data type: BOOL. TRUE: enable it; FALSE: disable i**t.**
Port:	Data type: DINT. COM port number (can be 1 to 37, depends on PAC). (*** Must use a constant value, cannot be a changed value. ***)
Addr:	Data type: DINT. The Net-ID address of the module , can be 1 to 255. (*** Must use a constant value, not a changed value ***)

Output Parameters:

OK:	Data type: BOOL. TRUE: Communication is Ok. FALSE: Communication failed.

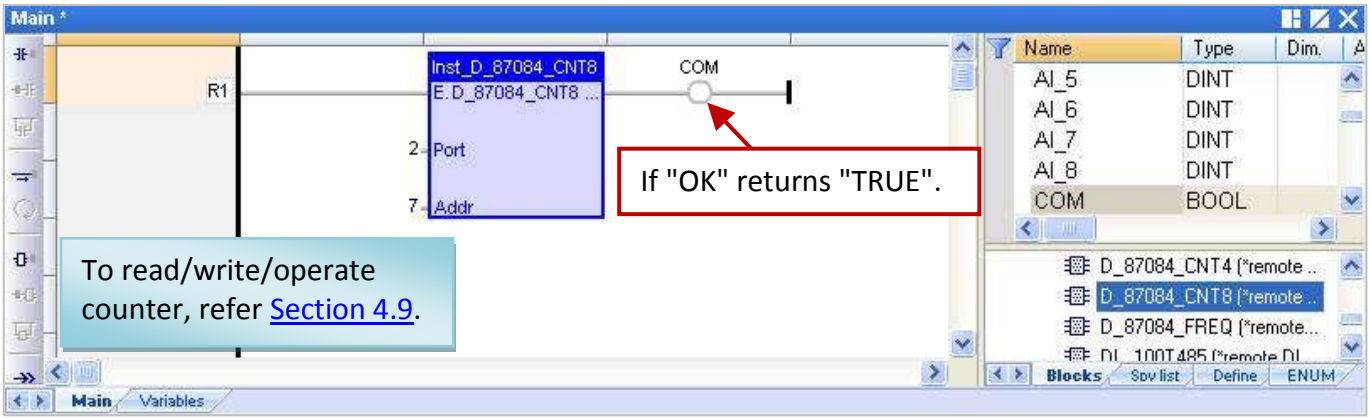
8.2.6 "D_87084_CNT8" Function Block

"D_87084_CNT8": Connect a remote I-87084W in an I/O Expansion Unit (e.g., I-87K4/5/8/9 or RU-87P4 or RU-87P8) to measure 8-ch counters.

Note:

1. Please MUST configure the I-87084W's counter data format as "**Hex format**" by DCON utility (see P8-1) when using the I-87084W to measure counters. Or it will not work.
2. Please use "DCON"(Section 8.1) in the "I/O boards" window and set proper settings (Port, baud_rate, etc.) on it.
3. To get the 8-ch counter value from the remote I-87084W, first using "D_87084_cnt8" Function Block. Then, using the "Counter_Start", "Counter_Stop", "Counter_Get", "Counter_State" and "Counter_Reset" Functions (refer Section 4.9) to operate counter channels.
4. All values of AI channels are meaningful only when the returned communication state is TRUE (If "OK" returns "TRUE".) .
5. Referring Chapter 12, click the menu bar "File" > "Add Existing Project" > "From Zip" to restore the demo project (CD-ROM: \Napdos\Win-GRAF\demo-project\DEMO_D_87084_C8.zip) in the shipping CD and see the program and descriptions.

Supposition: Use PAC's COM2 to connect the I-87084W (Addr. = 7) to measure 8-ch counters.



Input Parameters:

EN:	Data type: BOOL. TRUE: enable it; FALSE: disable i**t.**
Port:	Data type: DINT. COM port number (can be 1 to 37, depends on PAC). (*** Must use a constant value, cannot be a changed value. ***)
Addr:	Data type: DINT. The Net-ID address of the module , can be 1 to 255. (*** Must use a constant value, not a changed value ***)

Output Parameters:

OK:	Data type: BOOL. TRUE: Communication is Ok. FALSE: Communication failed.

8.2.7 "DL_100T485" Function Block

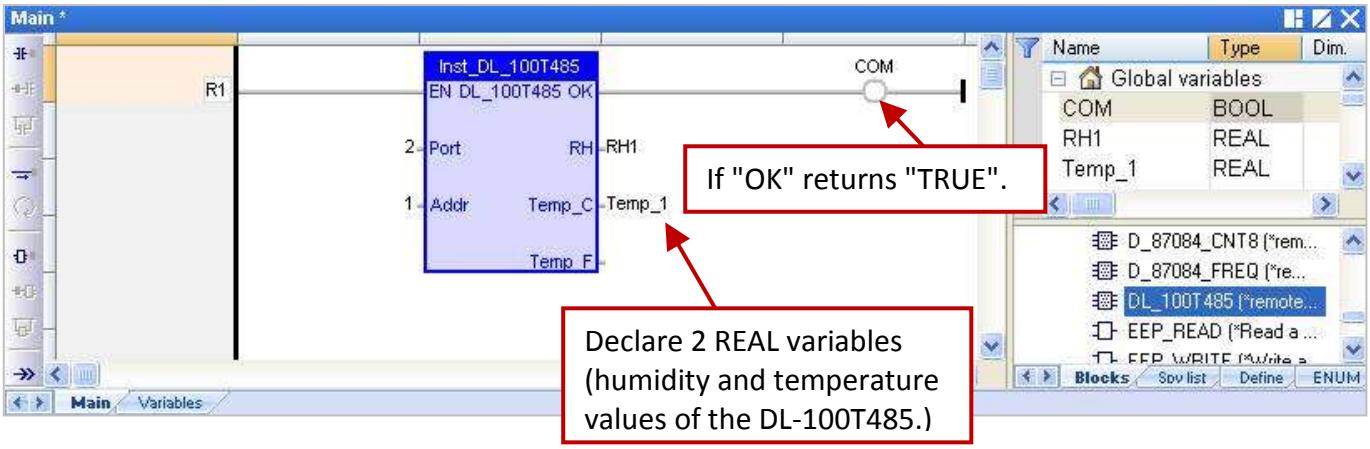
"DL_100T485": Connect a remote DL-100T485 module to get humidity and temperature value.

Product website: http://www.icpdas.com/root/product/solutions/remote_io/rs-485/dl_series/dl-100t485.html

Note:

1. Please use "DL-100T485 Utility" software in the shipping CD to configure the appropriate parameters of the module (e.g., Module ID). The TDL-100T485's default Address (ID) is "1", Baudrate is "9600", and the Checksum is "Disable".
2. Please use "DCON"(Section 8.1) in the "I/O boards" window and set proper settings (Port, baud_rate, etc.) on it.
3. All values of AI channels are meaningful only when the returned communication state is TRUE (If "OK" returns "TRUE".) .
4. Referring Chapter 12, click the menu bar "File" > "Add Existing Project" > "From Zip" to restore the demo project (CD-ROM: \Napdos\Win-GRAF\demo-project\ DEMO_DL_100T485.zip) in the shipping CD and see the program and descriptions.

Supposition: Use PAC's COM2 to connect the DL_100T485 (Addr. = 1) to get humidity and temperature value.



Input Parameters:

EN:	Data type: BOOL. TRUE: enable it; FALSE: disable it.
Port:	Data type: DINT. COM port number (can be 1 to 37, depends on PAC). (*** Must use a constant value, cannot be a changed value. ***)
Addr:	Data type: DINT. The Net-ID address of the module, can be 1 to 255. (*** Must use a constant value, not a changed value ***)

Output Parameters:

OK:	Data type: BOOL. TRUE: Communication is Ok. FALSE: Communication failed.
RH:	Data type: REAL. The value is "Relative humidity"; unit is 1%. For example, a value "45.7" means 45.7%.
Temp_C:	Data type: REAL. The temperature value is in "Degree Celsius". For example, a value "25.7" means 25.7 Degree C.
Temp_F :	Data type: REAL. The temperature value is in "Degree Fahrenheit". For example, a value "78.26" means 78.26 Degree F.

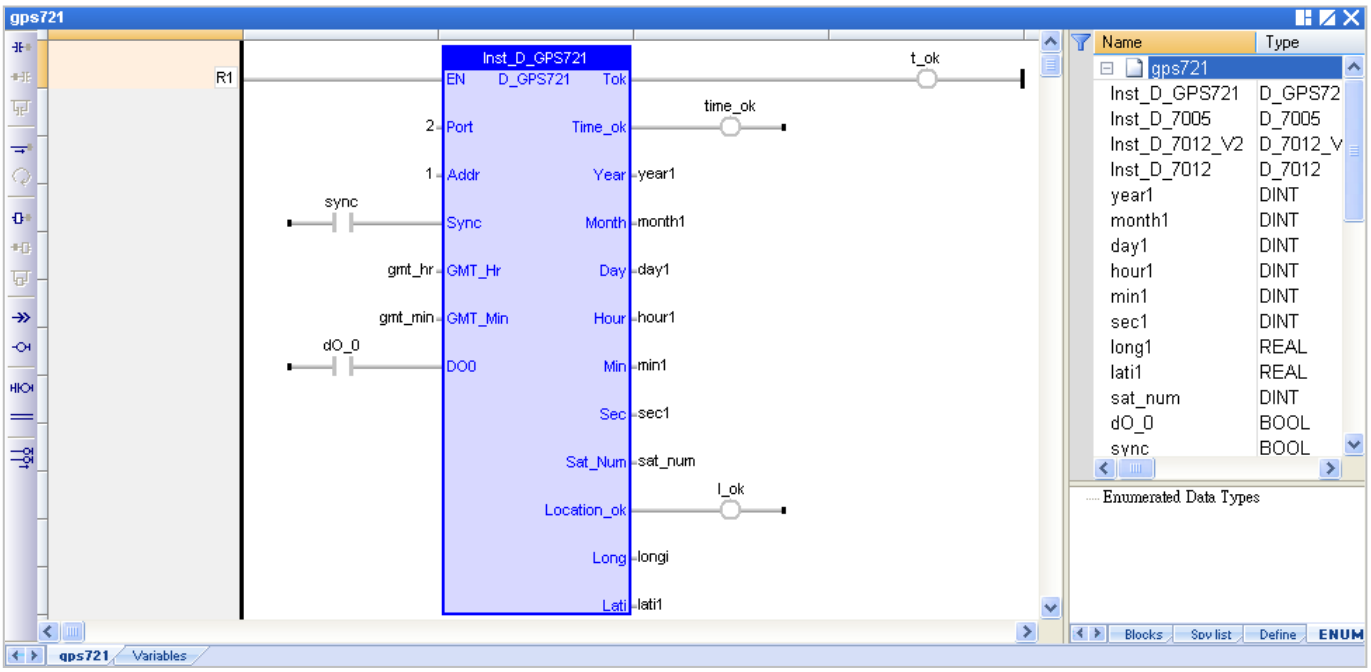
8.2.8 “D_GPS721” Function Block

Using the “D_GPS721” function to link one “GPS-721” remote GPS receiver module (includes one DO and one PPS outputs) which is used to receive GPS signals, get the precise date, time, longitude and latitude, and for satellite positioning and time correction. Moreover, “GPS-721” is equipped with a RS-232 interface, the remote host can request the GPS-721’s GPS information and remotely control the built-in DO channel by using DCON commands over RS-485. And, the PPS (Pulse Per Second) function can be used for a simple time synchronization.

Product website:
http://www.icpdas.com/root/product/solutions/industrial_wireless_communication/wireless_solutions/gps-721_tc.html

Note:

1. One PAC can use only one GPS-721 module.
2. All connected modules should be configured once. First of all, set up parameters of the GPS-721 module (e.g., Address, Baudrate, etc.) by the DCON Utility (see P8-1). (By defaults, the Address (ID) is "1", Baudrate is "9600", and Checksum is "Disable".)
3. Please add "DCON" (Section 8.1) in the "I/O boards" window and fill in the proper settings (Port, baud_rate, etc.).
4. Referring Chapter 12, click the menu bar "File" > "Add Existing Project" > "From Zip" to restore the demo project (CD-ROM: \Napdos\Win-GRAF\demo-project\dmeo_gps721.zip) in the shipping CD and see the program and descriptions.)



Input Parameters:

EN: Data type: BOOL. TRUE: enable it; FALSE: disable it.

Port: Data type: DINT. COM port number (can be 1 to 37, depends on PAC).
(*** Must use a constant value, cannot be a changed value. ***)

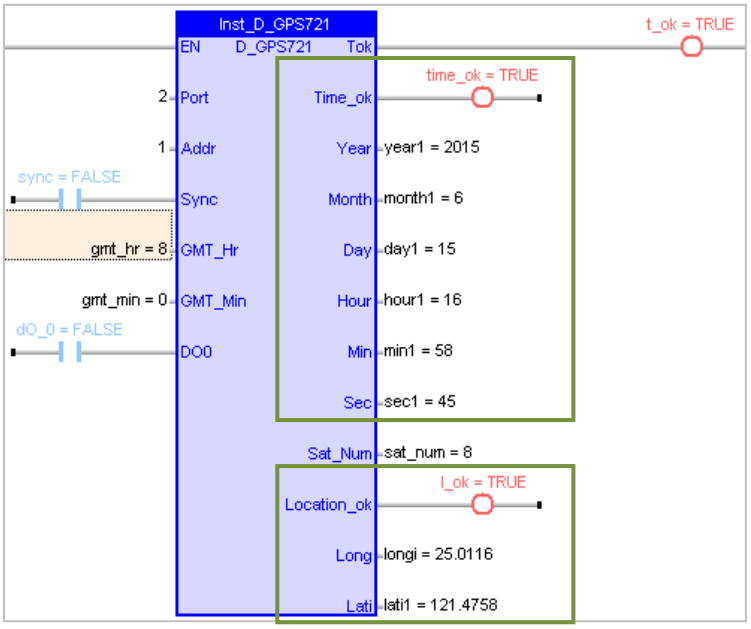
Addr: Data type: DINT. The Net-ID address of the module, can be 1 to 255.
(*** Must use a constant value, not a changed value ***)

Sync: Data type: BOOL. Set it as "TRUE" to enable auto time synchronization (**note:** It will work only when the "Time_ok" is also set to "TRUE"). If the time gap between the GPS-721 and the PAC is 5 seconds (or more), it will automatically correct the PAC time. Set it as "FALSE" to disable this function.

GMT_Hr & GMT_Min:

Data type: DINT. The time difference between local time and GMT (Greenwich Mean Time). For example, Beijing and Taipei are +8 hours (GMT_Hr=8, GMT_Min=0), United States is -6 hours (GMT_ Hr= -6, GMT_Min=0), and India is +5.5 hours (GMT_Hr=5, GMT_Min=30).

DO0: Data type: BOOL. The Digital output channel of the GPS-721 module.



Output Parameters:

Tok: Data type: BOOL.

TRUE: The communication of GPS-721 is normal.

FALSE: The communication of GPS-721 is failed. And all the following output values are invalid.

Time_ok: Data type: BOOL.

TRUE: Now, the values of Year, Month, Day, Hour, Min and Sec are valid.

FALSE: Now, the values of Year, Month, Day, Hour, Min and Sec are invalid. (i.e., an error or non-real-time value.)

Note: Each day from 23:59:00 to 00:00:59 (2 minutes), the "Time_ok" will be set as "FALSE" automatically and the time synchronization will not be processed.

Year: Data type: DINT. Year (2009 ~ ...).

Month: Data type: DINT. Month (1 ~ 12).

Day: Data type: DINT. Day (1 ~ 31).

Hour: Data type: DINT. Hour (0 ~ 23).

Min: Data type: DINT. Minute (0 ~ 59).

Sec: Data type: DINT. Second (0 ~ 59).

Sat_Num: Data type: DINT. The number of used satellites.
(0: no satellites can be found. Or, using 1 ~ 9 satellites)

Location_ok: Data type: BOOL.

FALSE: Now, the values of "Long" and "Lati" are invalid (i.e., an error or non-real-time value.)

TRUE: The GPS-721 has got the current longitude and latitude location.

(Only when "Location_ok" is set to "TRUE", the values of "Long" and "Lati" are valid.)

Long: Data type: REAL. Longitude (positive value: East ; negative value: West).
(For example, "25.0121" means 25.0121 degrees.)

Lati: Data type: REAL. Latitude (positive value: North ; negative value: South).
(For example, "121.4576" means 121.4576 degrees.)

Chapter 9 On Line Change

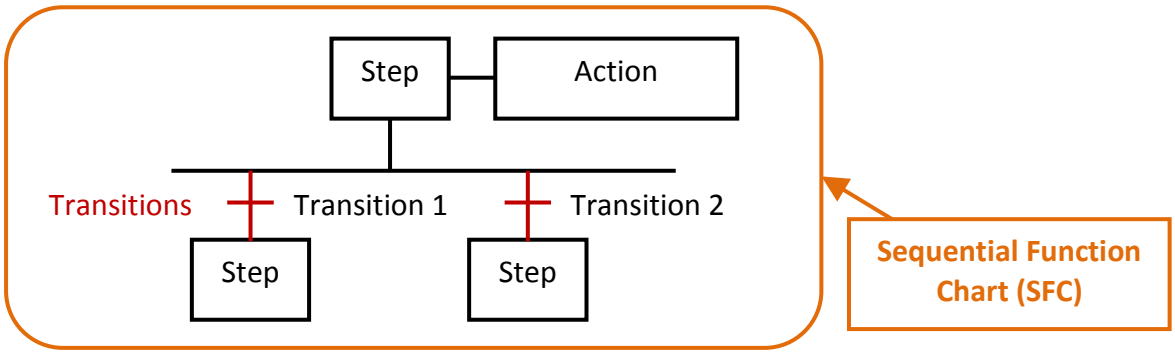
"On Line Change" function allows Win-GRAF PAC change its application to a modified one during the PAC running time. The modified application must be the same name as the original one that currently running on the PAC. The "On Line Change" is primarily used for emergency, such as, when the application is not allowed to be disabled or stopped for a while or cannot find time to replace a new application (for example, the device needs 24 hours operation and cannot be stopped). Except the above situations, it is not recommended to use this function! You had better stop the running application, then download the modified application to the PAC (see Section 2.3.5), which is more safe.

9.1 Limitations of "On Line Change"

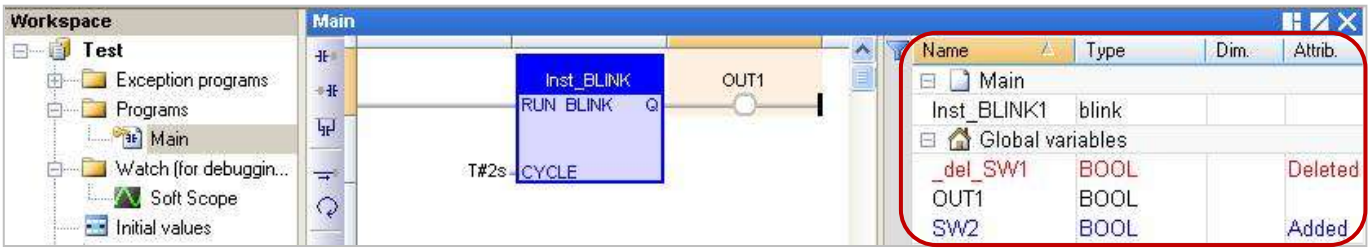
Please notice the important limitations before enabling the "On Line Change"!

When On Line change is enabled, you can perform on the fly the following kinds of changes:

- Change the code of a program.
- Change the condition of a SFC transition or the actions of a SFC step.

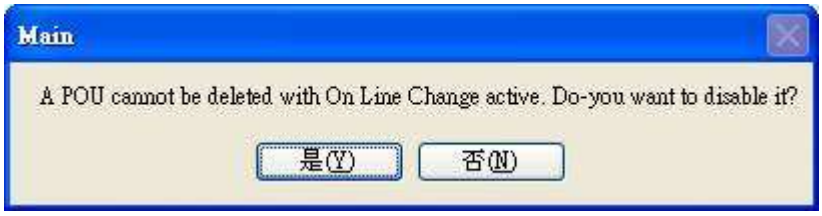


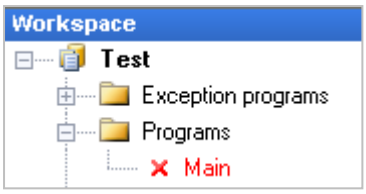
- Create, rename or delete global and local variables.
- Create, rename or delete global and local function block instances.



The following kinds of changes are not allowed:

- Create, delete or rename a program. (It will appear a warning message if delete a program.)





- Change SFC charts.
- Change the local parameters and variables of a UDFB.
- Change the type or dimension (or string length) of a variable or function block instance.
- Change the set of I/O boards.
- Change the definition of RETAIN variables.

In addition, the following programming features that are not safe during a change should not be used:

- Pulse (P or N) contacts and coils (edge detection).

  ✍ Instead, you must use declared instances of R_TRIG and F_TRIG function blocks.

Rising Pulse Detection		
	Before Enable	**After Enable**
P (False > True)	SW1 P	SW1 Inst_R_TRIG CLK R_TRIG Q
	OUT1 P	Inst_R_TRIG CLK R_TRIG Q OUT1
Decreased Pulse Detection		
N (True > False)	SW1 N	SW1 Inst_F_TRIG CLK F_TRIG Q
	OUT1 N	Inst_F_TRIG CLK F_TRIG Q OUT1

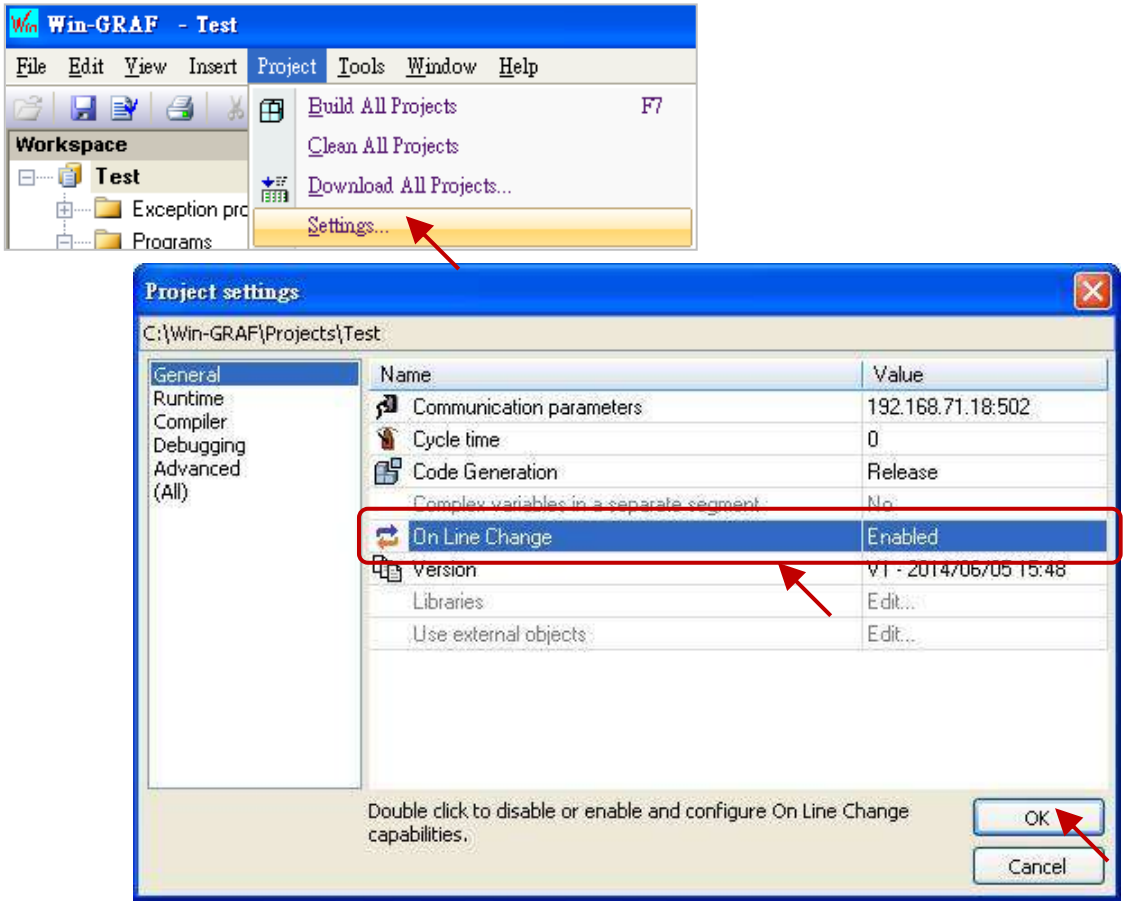
- Loops in FBD with no declared variable linked.

  ✍ You need to explicitly insert a variable in the loop.

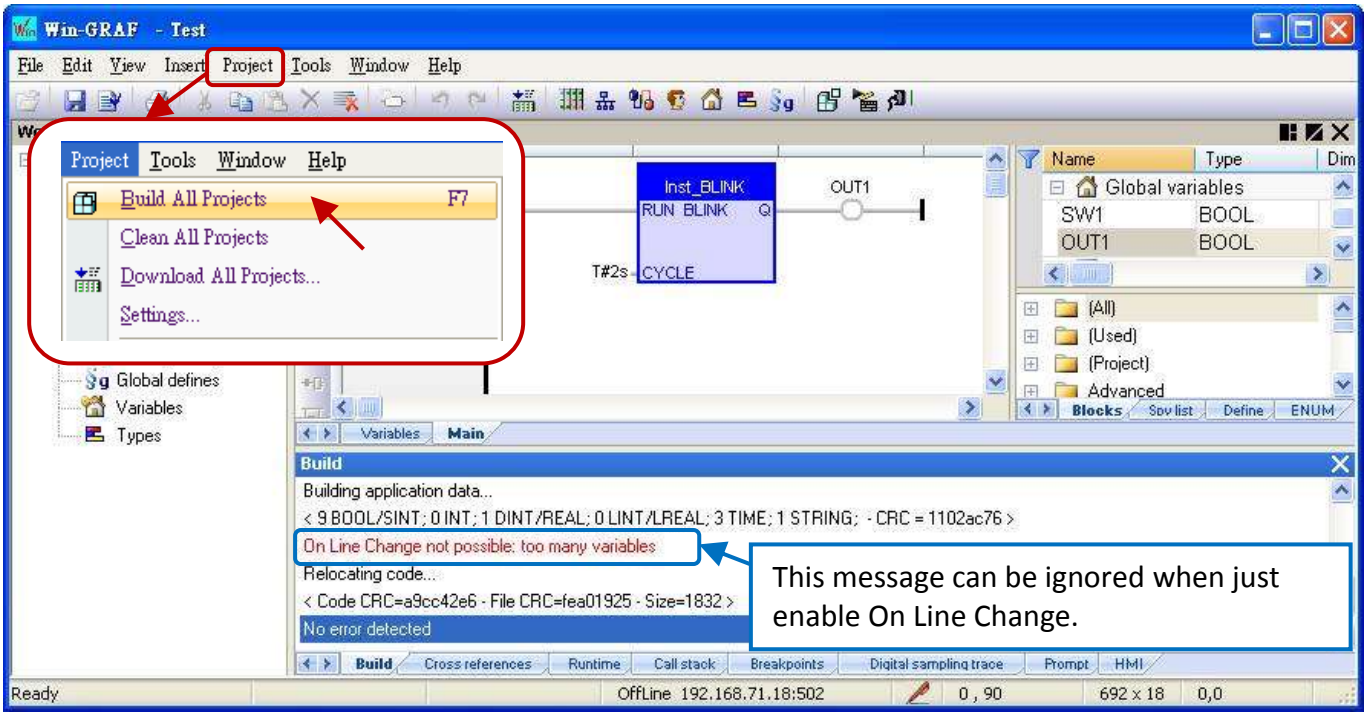
9.2 Using "On Line Change"

Enable The "On Line Change" Function:

1. Mouse click "Project > Settings..." from the menu bar, then double click "On Line Change" item to set it "Enabled".



2. Click "Project > Build All Projects" from the menu bar. Must compile the program first, then can set up the following steps.



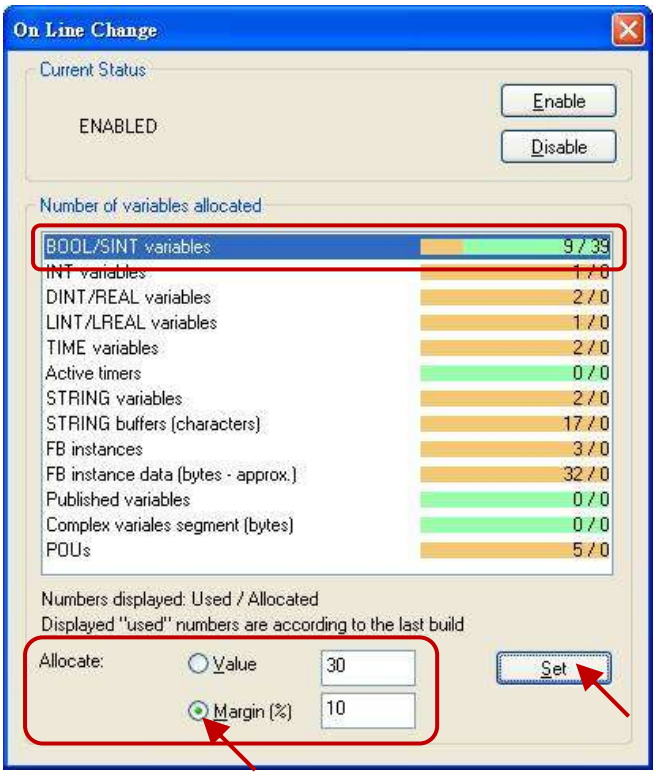
Setup the Number of Variables:

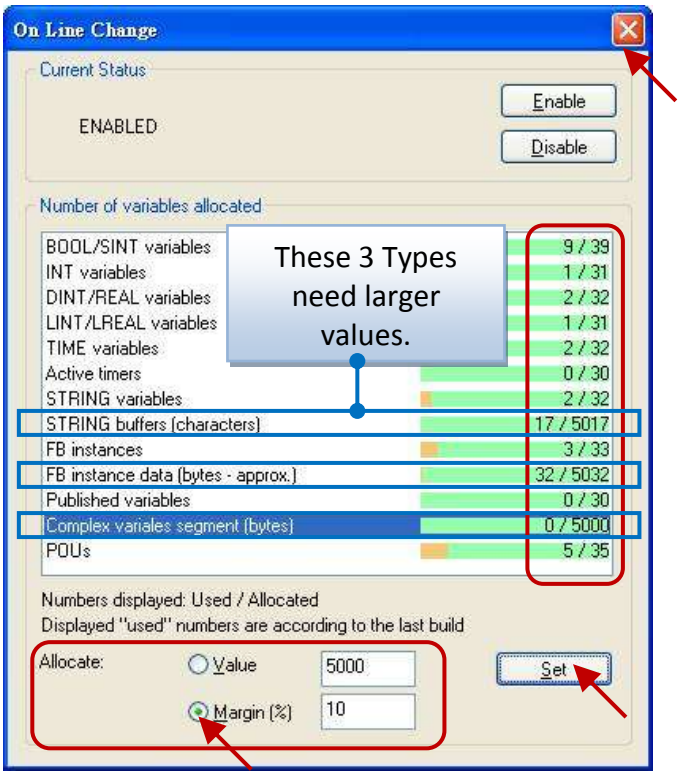
When the "On Line Change" is enabled, you have to set up the number of variables reserved for the declaration of the new variables and Function Blocks for future on-line change usage.

3. The same as the step 1, mouse click "Project > Settings..." from the menu bar, then double click on the "On Line Change" to show the setting window.
   Please set the needed new number in the “Value" or “Margin" fields.
   Note: If both “Value" and “Margin" have set values, it will use the larger value. In this example, "Value" is set as "30" and “Margin" is set as "10", then the displayed value x 10% is smaller than 30, so it will use the larger value "30".

4. Click the needed Variable Type, then click the "Set“ button.
   (E.g., Click "BOOL/SINT variables" and "Set" button, then the number become 9 + 30 = 39.)
   Note: "STRING buffers (characters)", "FB instance data (bytes – approx.)" and "Complex variables segment (bytes) " need to set a larger number (This example uses "5000").

5. After setting (as the picture below), click the "X" in the upper right corner to exit the setting.

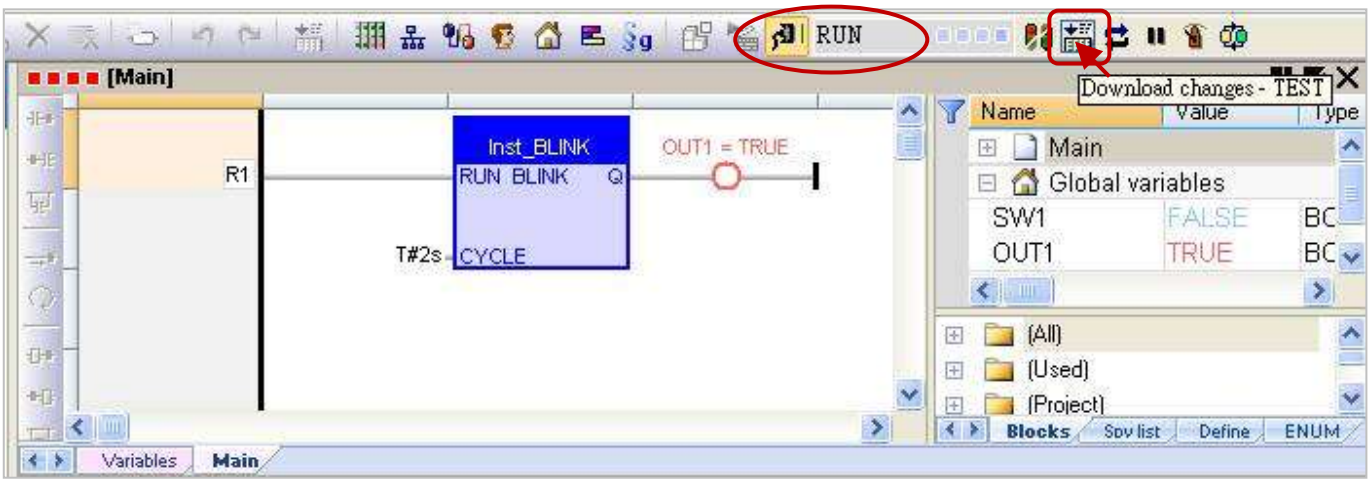




6. Click "Project > Build All Projects" from the menu bar, compile the program again. Then click “On Line” or click the tool icon to link to the PAC. (Refer the Section 2.3.5.)

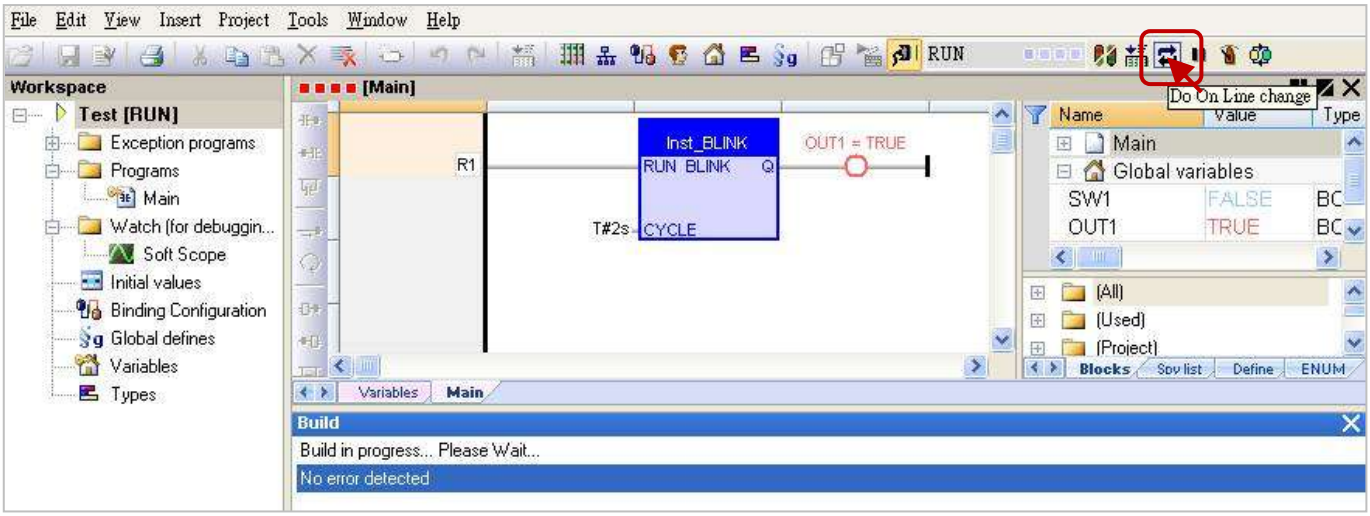
7. After successfully link to the PAC, click the tool icon "Download changes" to download the program to the PAC.

Note: "On Line Change" is only suitable for the program that just has a little change (Do not need to stop the application). If the program name is different from the running one on the PAC, the user has to stop and download the program again (refer the Appendix B).



8. Click on the tool icon "Do On Line change" to execute the program.

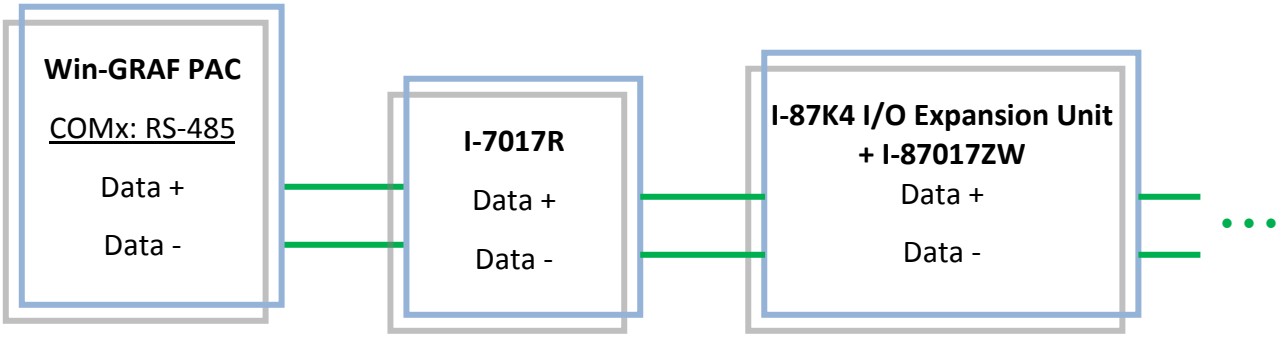
Note: After executing the "On Line Change", there are some using restrictions for protecting the system normal operation (refer the Section 9.1). So, make sure the program is correct, then perform this function.



Chapter 10 Data/Type Conversion and Using the PAC Time

10.1 AI Data Conversion

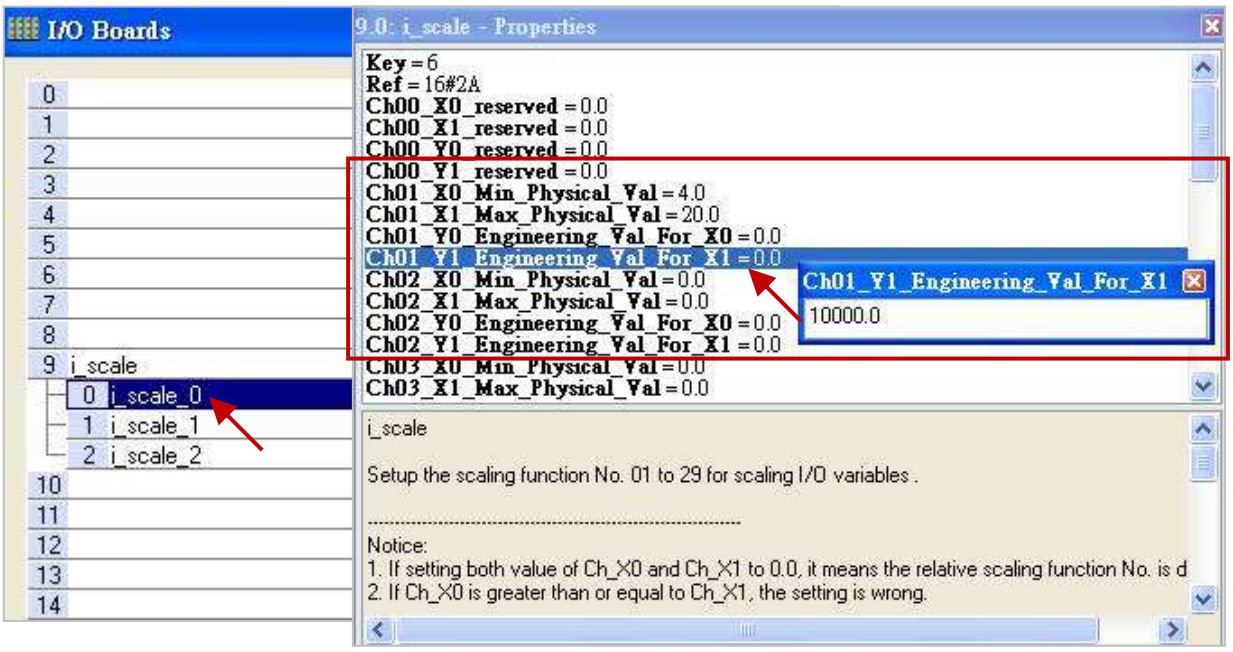
If you are using AI modules in the PAC's Slot 0 to 7 (for example: I-8017HW), and want to convert the AI input signal (for example: "4 to 20 mA" or "0 to 10 V") to the user engineering value (for example: 0 to 10000), refer the Section 4.3. However, if you are using the remote AI modules (e.g., through PAC's RS-485 Port to connect to the I-87017ZW or I-7017R module), you can refer the following settings:



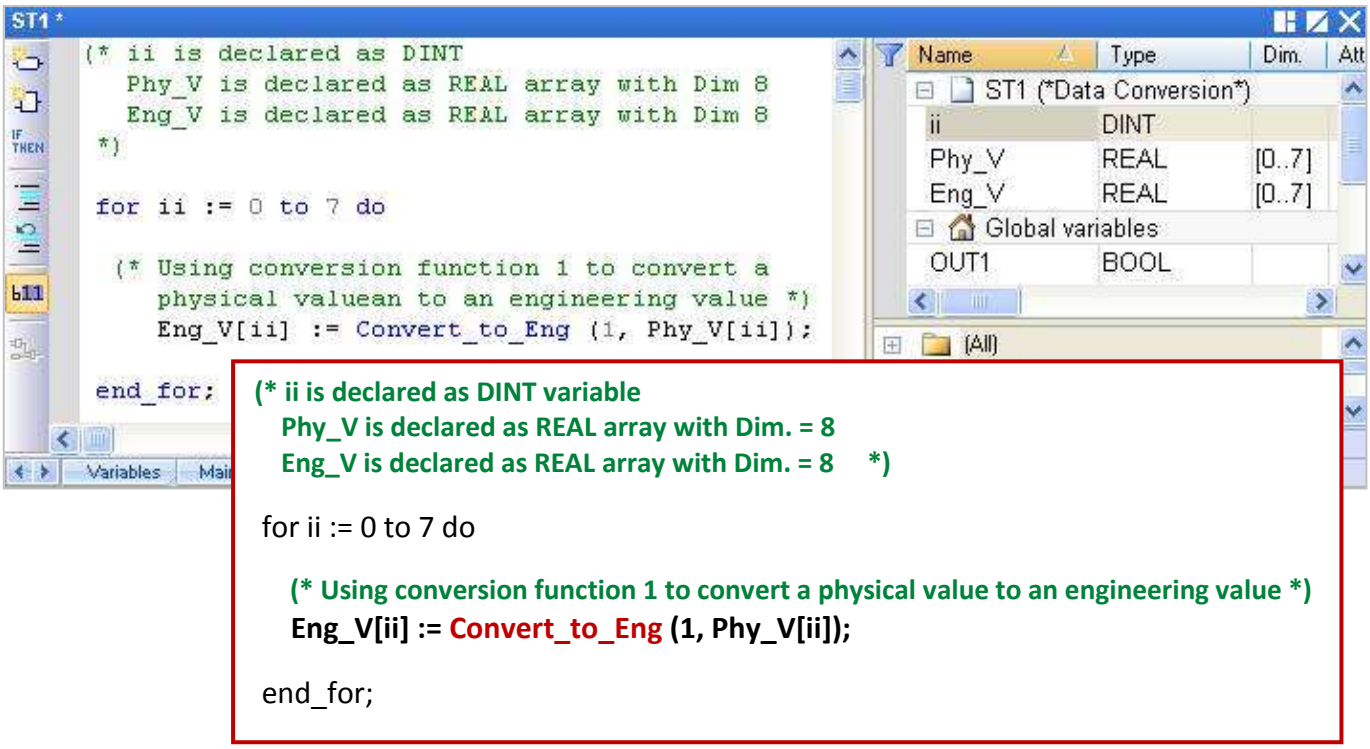
1. First, connect "i_scale" in the "I/O Boards" window, and double click on "i_scale_x" to open the Properties window (refer the Section 4.2 for detail steps)
   Note: "I/O Boards" supports ONLY ONE "i_scale" (DO NOT connect 2 or more "i_scale").

2. Set up the number and value of the Conversion Function that need to be enabled.
   (E.g., Use Function 1 to convert "4 to 20 mA" into "0 to 10000").

Ch**01**_X0_Min_Physical_Val:	"4.0"
Ch**01**_X1_Max_Physical_Val:	"20.0"
Ch**01**_Y0_Engineering_Val_For_X0:	"0.0"
Ch**01**_Y1_Engineering_Val_For_X1:	"10000.0"



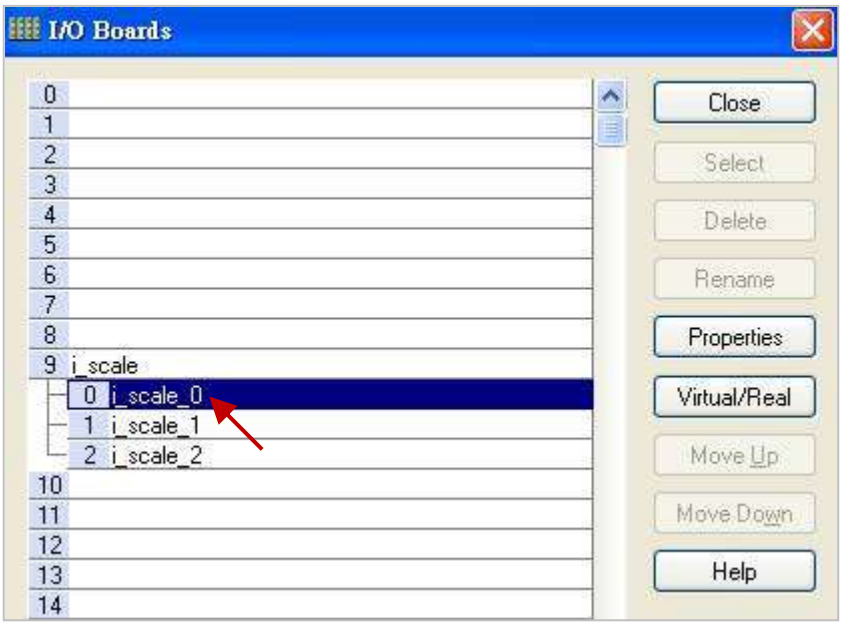
3. Edit an ST Program to convert a physical value (e.g., Phy_V[0] to [7]) to an engineering value (e.g., Eng_V[0] to [7]). (Refer the Section 2.3.3 for detail setting steps.)



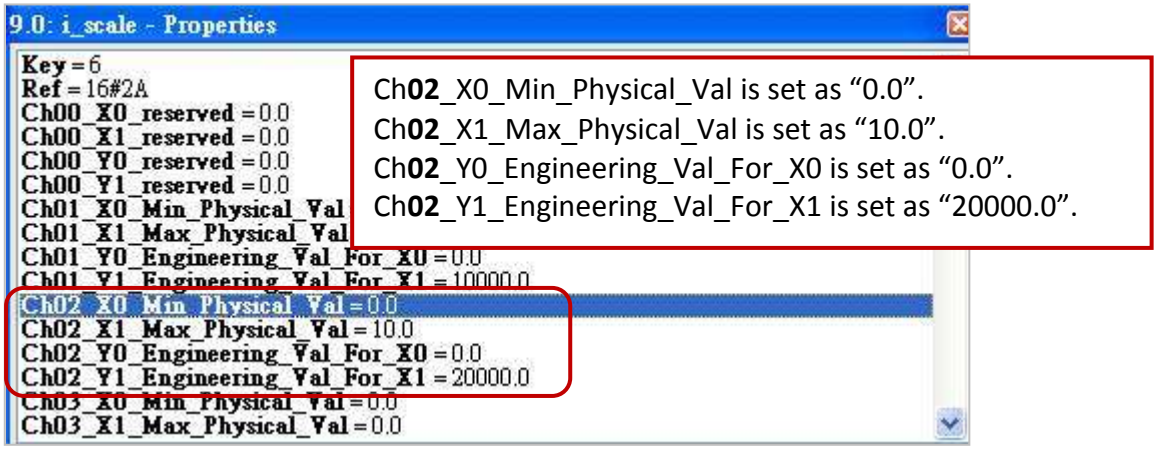
10.2 AO Data Conversion

If you are using AO modules in the PAC's Slot 0 to 7 (for example: I-8024W), and want to convert the engineering value to the AO output signal (e.g., convert "0 to 20000" to "0 to 10 V"), refer the Section 4.4. However, if you are using the DCON remote AO modules (e.g., through PAC's RS-485 Port to connect to the I-7024 module), you can refer to the following settings:

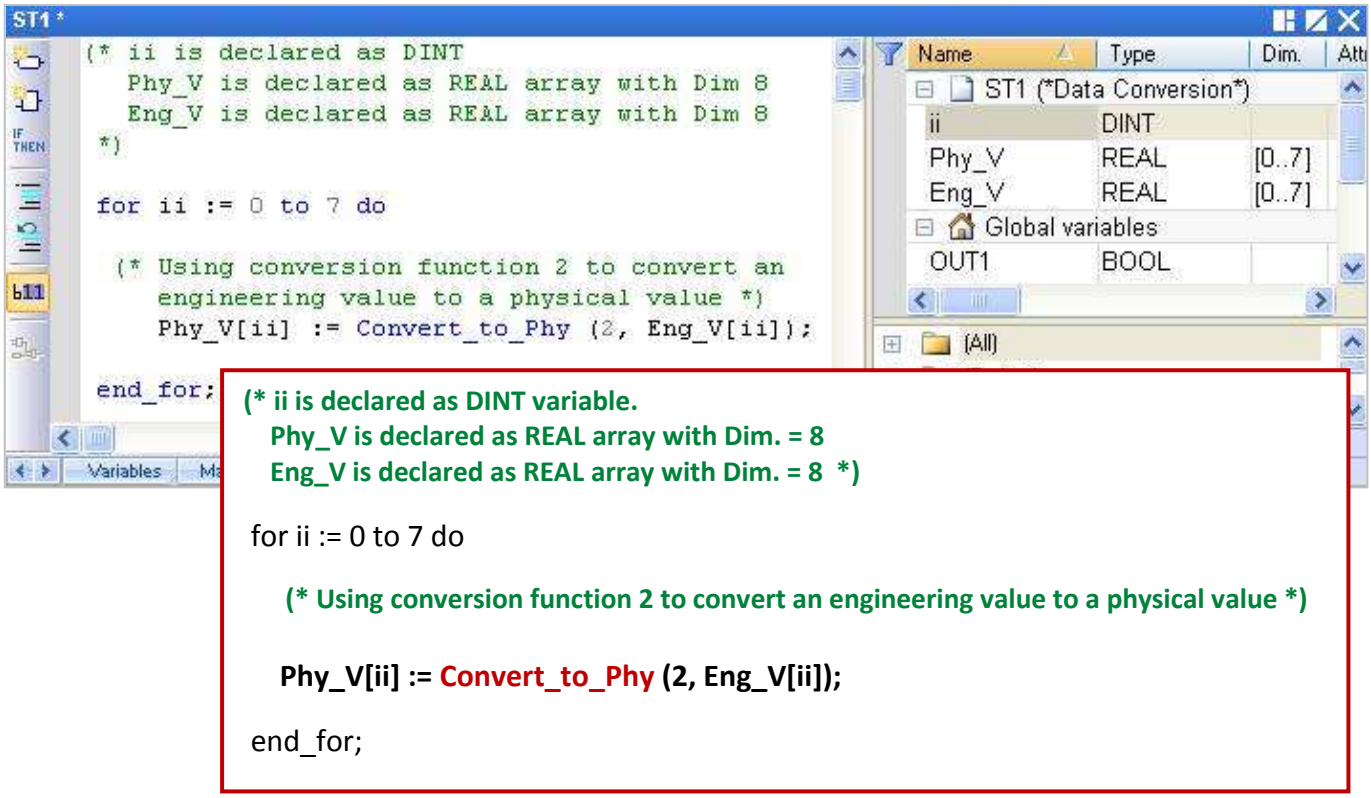
1. Connect "i_scale" in the "I/O Boards" window, and double click "i_scale_x" to open the Properties window (refer the Section 10.1 for detail steps)
   Note: "I/O Boards" supports ONLY ONE "i_scale" (DO NOT connect 2 or more "i_scale").



2. Set up the number and value of the Conversion Function (e.g., use Function 2 to convert "0 to 20000" to "0 to 10 V").



3. Edit an ST Program to convert an engineering value to a physical value (e.g., Eng_V[0] to [7]). (Refer the Section 2.3.3 for detail setting steps.)



10.3 Data Type Conversion

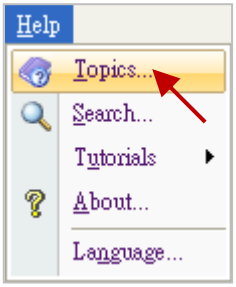
When different types of data want to do "+, -, *, /" calculation or ">, <, =, <=, > =, <> (not equal)" operation, or when the variable types of the parameters in the function are different, you must first use the type conversion function (in the following table) to convert them into the same data type, then can use the data normally.

Data Conversion Functions	Descriptions	Data Conversion Functions	Descriptions
ANY_TO_BOOL	Convert to Boolean	ANY_TO_REAL	Convert to Real
ANY_TO_SINT	Convert to Short Integer (8-bit)	ANY_TO_LREAL	Convert to Double
ANY_TO_INT	Convert to Integer (16-bit)	ANY_TO_STRING	Convert to String
ANY_TO_DINT	Convert to Long Integer (32-bit – Default)	NUM_TO_STRING	Convert Number to String. Can set the decimal digital number after converting
ANY_TO_LINT	Convert to Large Integer (64-bit)	ATOH	Convert Hexadecimal String to Integer
ANY_TO_TIME	Convert to Timer	HTOA	Convert Integer to Hexadecimal String

For example, the following ST program will convert the DINT variable to a Real first, and then do the calculation.

```
REAL_Val_1 := ANY_TO_REAL (DINT_Val_1) * 3.5 + 4.8 ;
```

You can open the "HTML Help" from the menu bar, and enter the searching key word you want to see the detail setup instructions.



10.4 BCD Conversion

BCD 4-bit code is used to represent decimal numbers from 0 to 9. Suppose there is a decimal value "132", if converted to BCD code is "000100110010" and if converted to binary is 10000100 (e.g., $2^7 + 2^2 = 128 + 4 = 132$).

Decimal	BCD				Description
	2^3	2^2	2^1	2^0	
0	0	0	0	0	0
1	0	0	0	1	$2^0 = 1$
2	0	0	1	0	$2^1 = 2$
3	0	0	1	1	$2^1 + 2^0 = 3$
4	0	1	0	0	$2^2 = 4$
5	0	1	0	1	$2^2 + 2^0 = 5$
6	0	1	1	0	$2^2 + 2^1 = 6$
7	0	1	1	1	$2^2 + 2^1 + 2^0 = 7$
8	1	0	0	0	$2^3 = 8$
9	1	0	0	1	$2^3 + 2^0 = 9$

NOTE:
BCD code can only represent numbers from 0 to 9.
It can not be used for these six values (1010, 1011, 1100, 1101, 1110, 1111), and will return "0".

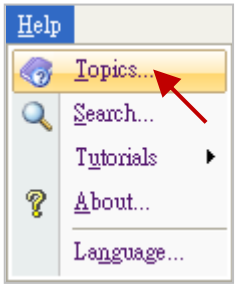
The function table below can be used for BCD (Binary Coded Decimal) value conversion.

Type Conversion Function	Description
BIN _TO_BCD	Convert Binary to BCD value
BCD_TO_BIN	Convert BCD value to Binary

BIN _TO_BCD:
19$_{(10)}$ = 0001 1001$_{(BCD)}$
= $2^4 + 2^3 + 2^0$ = 16 + 8 + 1 = **25**

BCD_TO_BIN:
33$_{(10)}$ = 0010 0001$_{(2)}$ = **21**$_{(BCD)}$
15$_{(10)}$ = 0000 ~~1111~~$_{(2)}$ = **0**$_{(BCD)}$

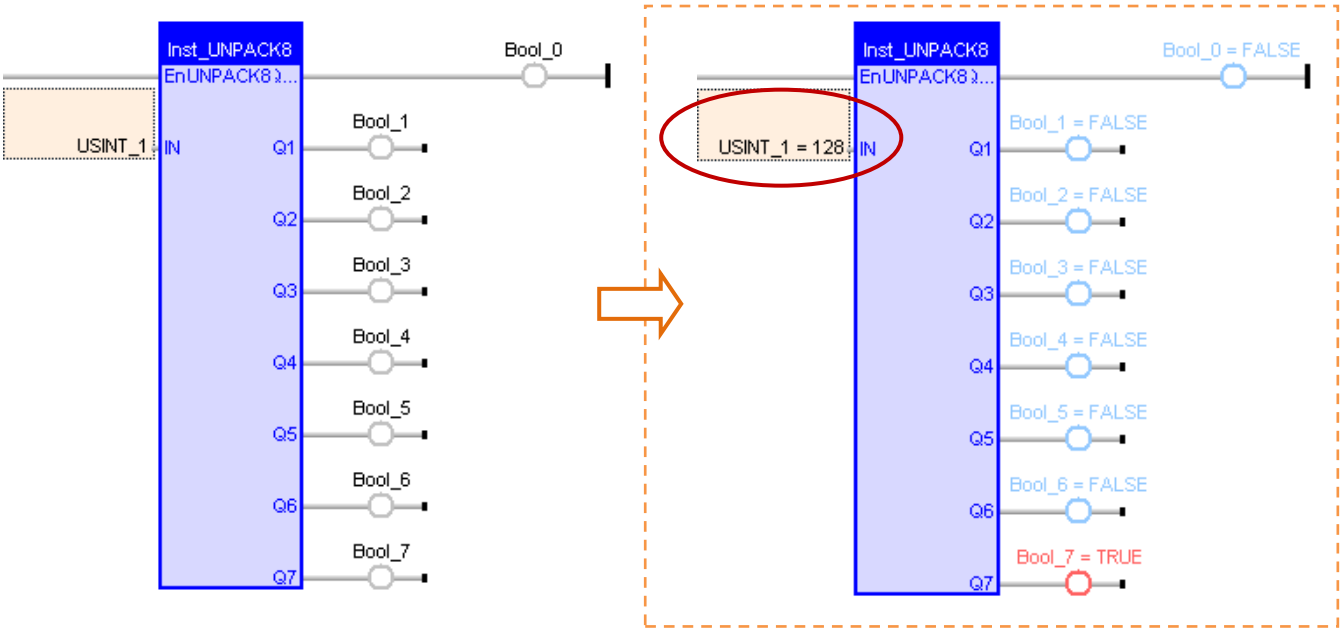
You can open the "HTML Help" from the menu bar, and enter the searching key word you want to see the detail setup instructions.



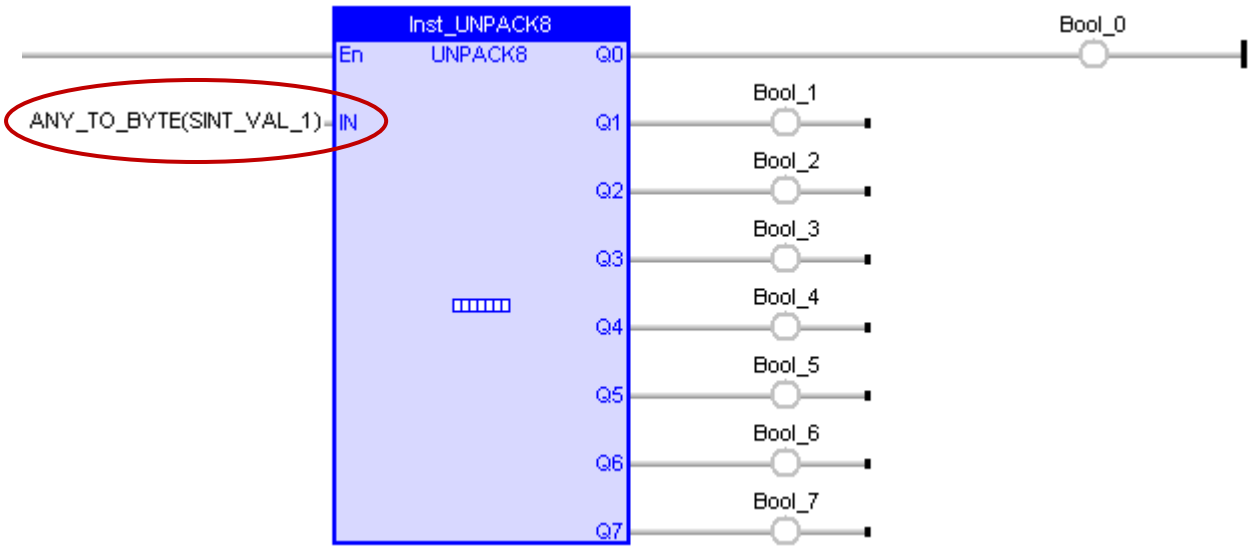
10.5 Pack/Unpack Integer or Boolean

Unpack Integer to Boolean:

If want to unpack one BYTE (or USINT, range: 0 to 255) to 8 Booleans, you can use "UNPACK8" Function Block.



If want to unpack one SINT to 8 Booleans, you must first use the ST program "ANY_TO_BYTE ()" to convert the SINT to be a BYTE type, as follows:



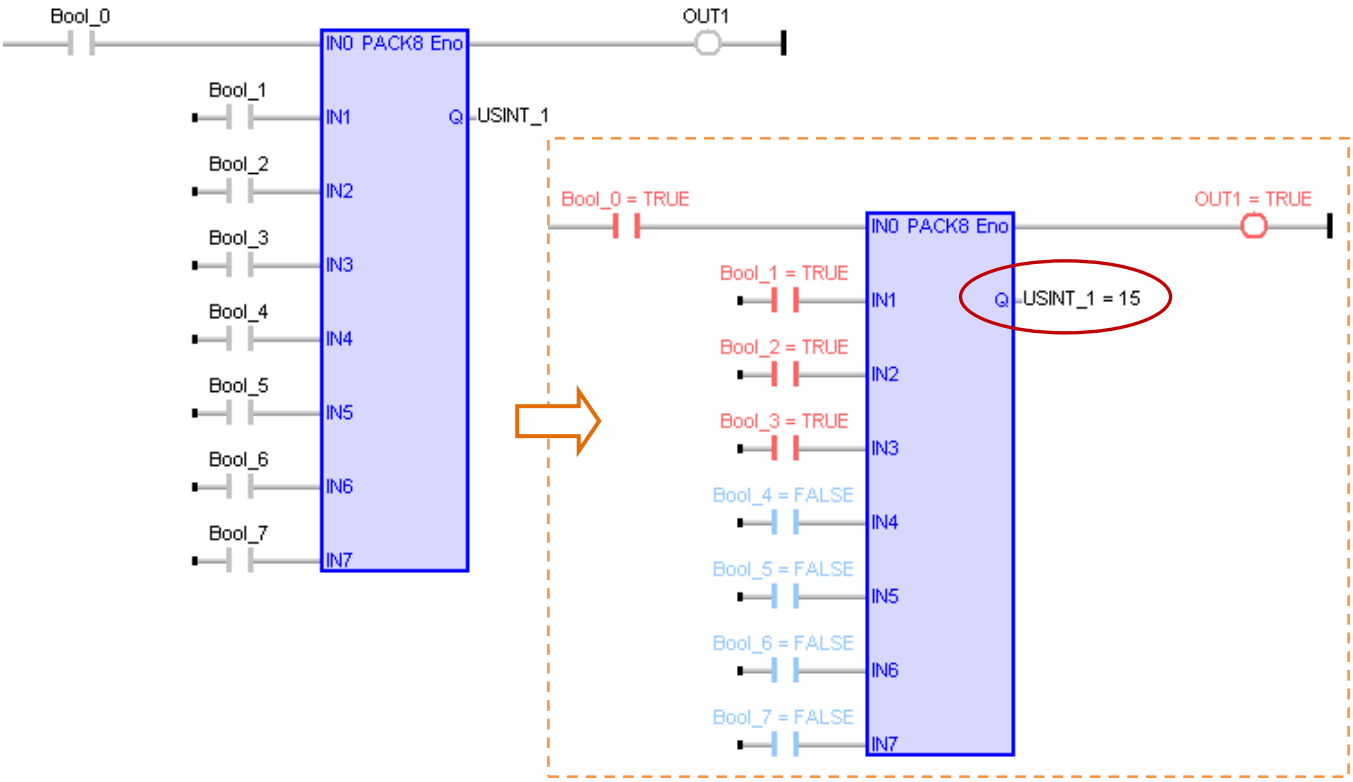
Pack Boolean Into Integer:

If want to pack 8 Booleans into one BYTE (or USINT, range: 0 to 255), you can use "PACK8" Function.

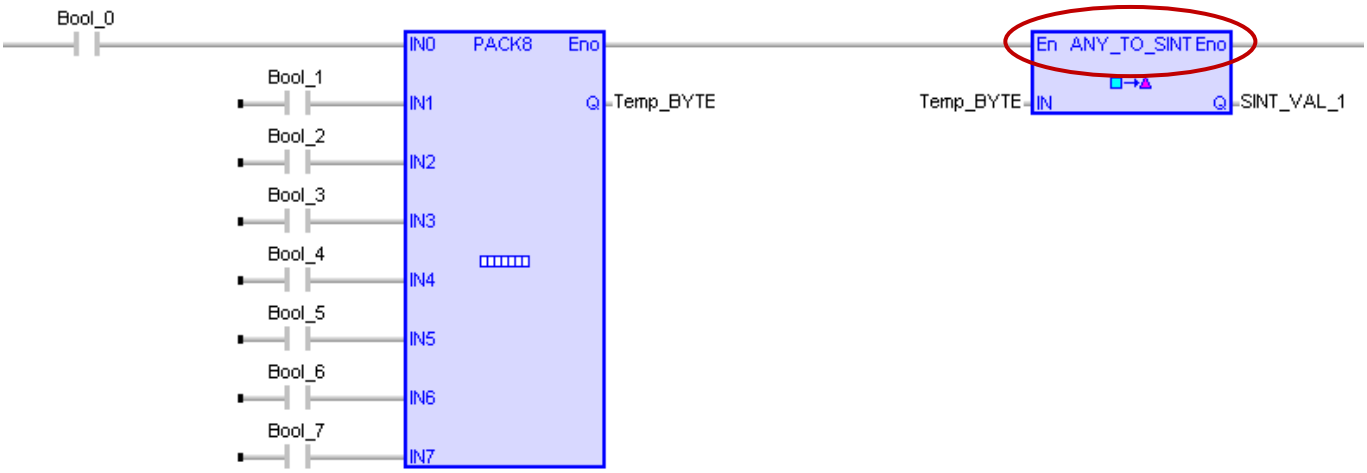
ST Program:

```
USINT_1  :=   PACK8 (Bool_0, I Bool_1, Bool_2, Bool_3, Bool_4, Bool_5, Bool_6, Bool_7);
```

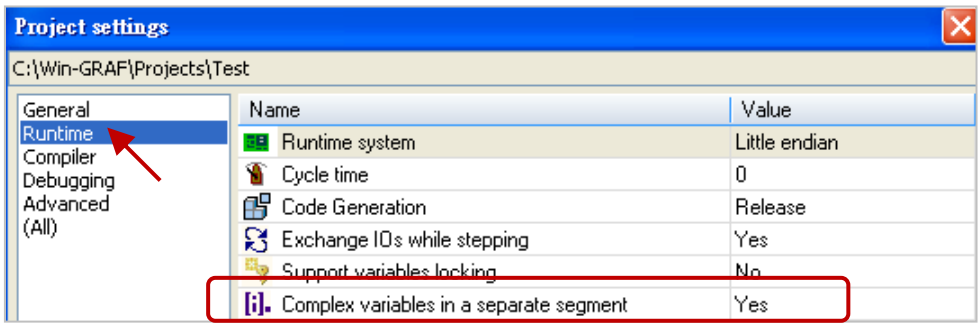
LD Program:



If want to pack 8 Booleans into one SINT, you must assign a "BYTE" variable to the output(Q) to save the value temporary, and use a "ANY_TO_SINT" Function to convert BYTE into SINT type, as follows:



Note: If the compiling fails, please click "Project" > "Settings" from the menu bar to check if the setting of "Complex variables in a separate segment" in the "Runtime" is "Yes".



10.6 Pack/Unpack BYTE, WORD, DWORD

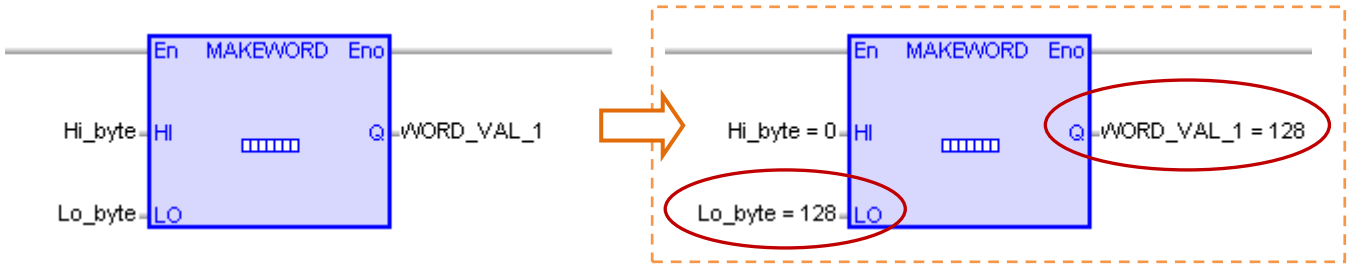
Pack Two 8-bit Data into One 16-bit Data

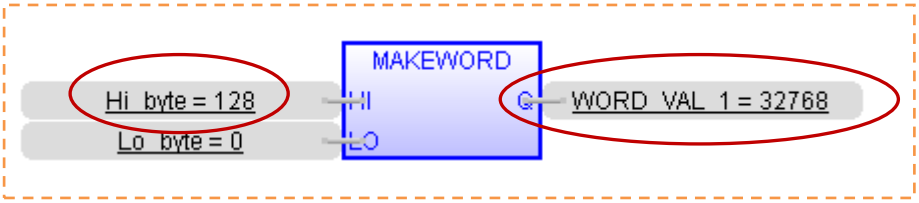
If want to pack 2 BYTE (or USINT) into one WORD (or UINT), you can use a "MAKEWORD" Function.

ST Program:

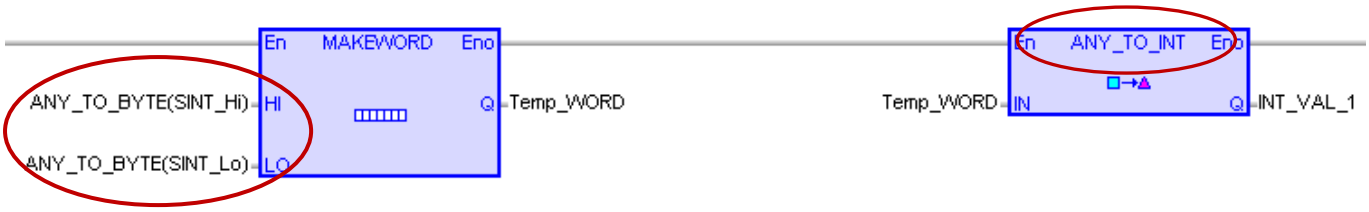
```
WORD_VAL_1  :=   MAKEWORD (Hi_byte, Lo_byte);
```

LD/FBD Program:





If want to pack 2 SINT into one INT, you must first use an ST program "ANY_TO_BYTE ()" to convert SINT into BYTE, and then use an "ANY_TO_INT" Function to convert the packed WORD into INT type.



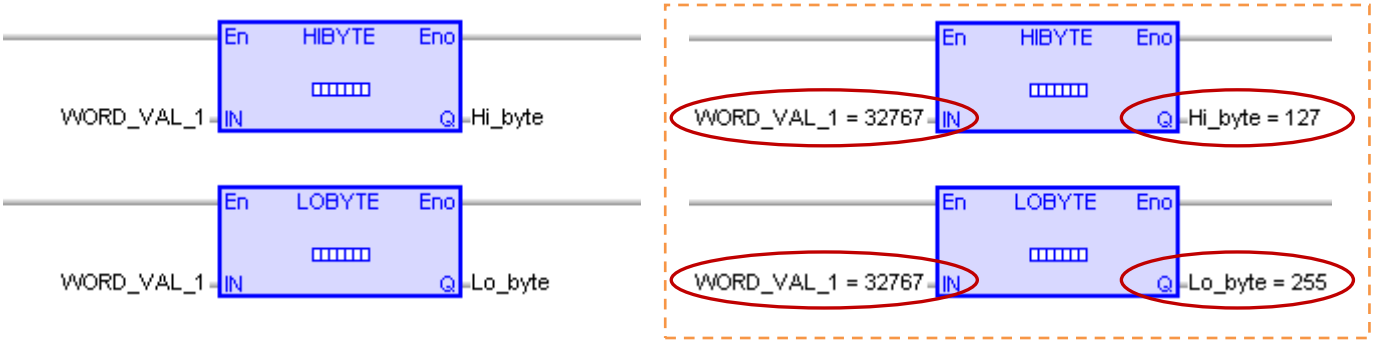
Unpack One 16-bit Data to Two 8-bit Data

If want to unpack one WORD (or UINT) to 2 Byte (or USINT), you can use "HIBYTE", "LOBYTE" Functions.

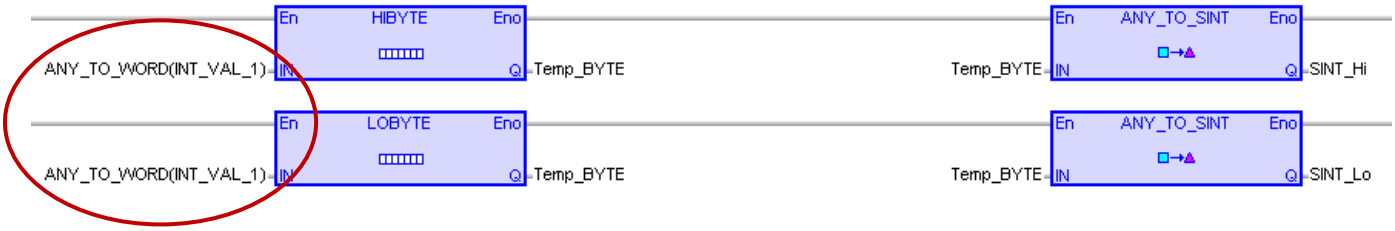
ST Program:

```
Hi_byte  :=   HIBYTE (WORD_VAL_1);
Lo_byte  :=   LOBYTE (WORD_VAL_1);
```

LD/FBD Program:



If want to unpack one INT to 2 SINT, you must first use an ST program "ANY_TO_WORD()" to convert INT into WORD, and then use an "ANY_TO_SINT" Function to convert the unpacked BYTE into SINT type.



Pack Two 16-bit Data into One 32-bit Data

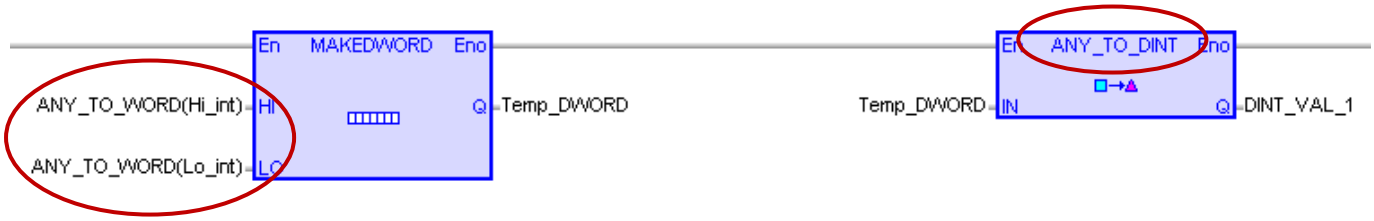
If want to pack 2 WORD (or UINT) into a DWORD (or UDINT), you can use a "MAKEDWORD" Functions.

ST Program:

```
DWORD_VAL_1 := MAKEDWORD (Hi_word, Lo_word);
```

LD/FBD Program:

If want to pack 2 INT into 1 DINT, you must first use an ST program "ANY_TO_WORD()" to convert INT to WORD, and then use an "ANY_TO_DINT" Function to convert the unpacked DWORD into DINT type.



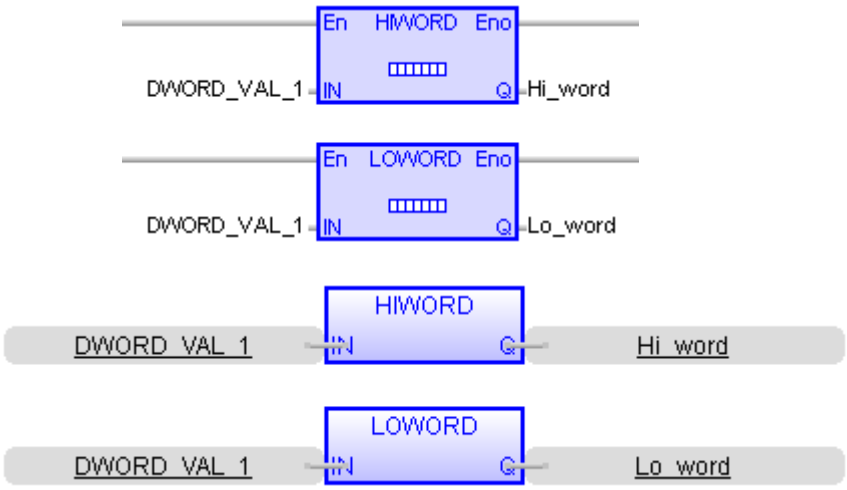
Unpack One 32-bit Data to Two 16-bit Data

If want to unpack one DWORD (or UDINT) to 2 WORD (or UINT), you can use "HIWORD", "LOWORD" Function Blocks.

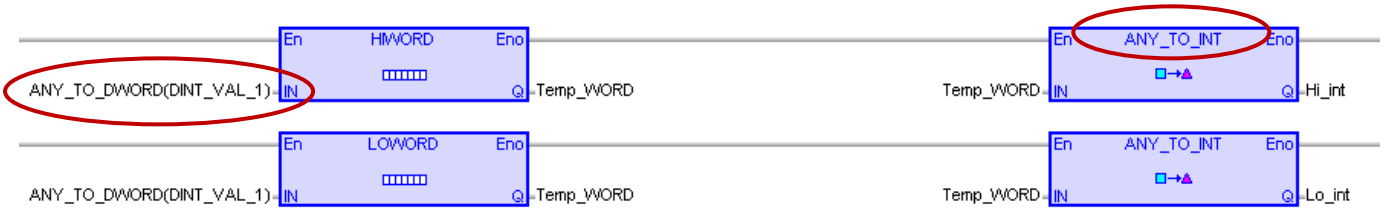
ST Program:

```
Hi_word  :=  HIWORD (DWORD_VAL_1);
Lo_word  :=  LOWORD (DWORD_VAL_1);
```

LD/FBD Program:



If want to unpack one DINT to 2 INT, you must first use an ST program "ANY_TO_DWORD()" to convert DINT to DWORD, and then use an "ANY_TO_INT" FB to convert the unpacked WORD into INT type.

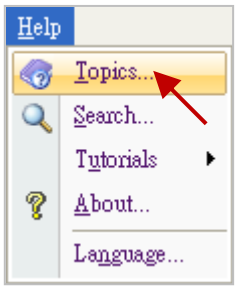


10.7 Unpack Variable to Byte Array or Pack Byte Array into Variable

"SerializeOut" Function can unpack a Win-GRAF Variable value to a Byte Array (or USINT Array);
"SerializeIn" Function can pack a Byte Array (or USINT Array) into a Win-GRAF Variable value.

Note: 1. The Dim. of Array must be set as at least "8".
2. This "Serialize" Function can not use the STRING variable.

You can open the "HTML Help" from the menu bar, and enter the searching key word to see the detail setup instructions.



If the SerializeOut() and SerializeIn() return "0", it means the saving location is wrong or the Array's space is not enough.

(* Declare TMP_DINT as a DINT,
buf as a BYTE Array, Dim. = 10,
DINT_Val as a DINT,
Word_Val as a WORD,
REAL_Val as a REAL *)

Note:

Data Type	Byte
BOOL, SINT, USINT, BYTE	1
INT, UINT, WORD	2
DINT, UDINT, DWORD, REAL	4
LINT, LREAL	8

Example 1

(* To unpack one DINT_Val to 4 Bytes, and save them separately to the buf[2], buf[3], buf[4] and buf[5] from the location 2 of the BYTE Array in the "Little Endian" sequence. *)

```
TMP_DINT := SerializeOut (buf, DINT_Val, 2, FALSE) ;
```

Note: The last parameter is "FALSE", that means to use the "Little Endian" sequence (To save the Low Byte to the starting address).

Example 2

(* To unpack one Word_Val to 2 Bytes, and save them separately to the buf[0] and buf[1] from the location 0 of the BYTE Array in the "Big Endian" sequence. *)

```
TMP_DINT := SerializeOut (buf, Word_Val, 0, TRUE) ;
```

Note: The last parameter is "TRUE", that means to use the "Big Endian" sequence (To save the High Byte to the starting address).

Example 3

(* To pack the buf[0], buf[1] , buf[2] and buf[3] in the BYTE Array into one REAL_Val in the "Little Endian" sequence. *)

```
TMP_DINT := SerializeIn (buf, REAL_Val, 0, FALSE) ;
```

Note: The last parameter is "FALSE", that means to use the "Little Endian" sequence (To save the Low Byte to the starting address).

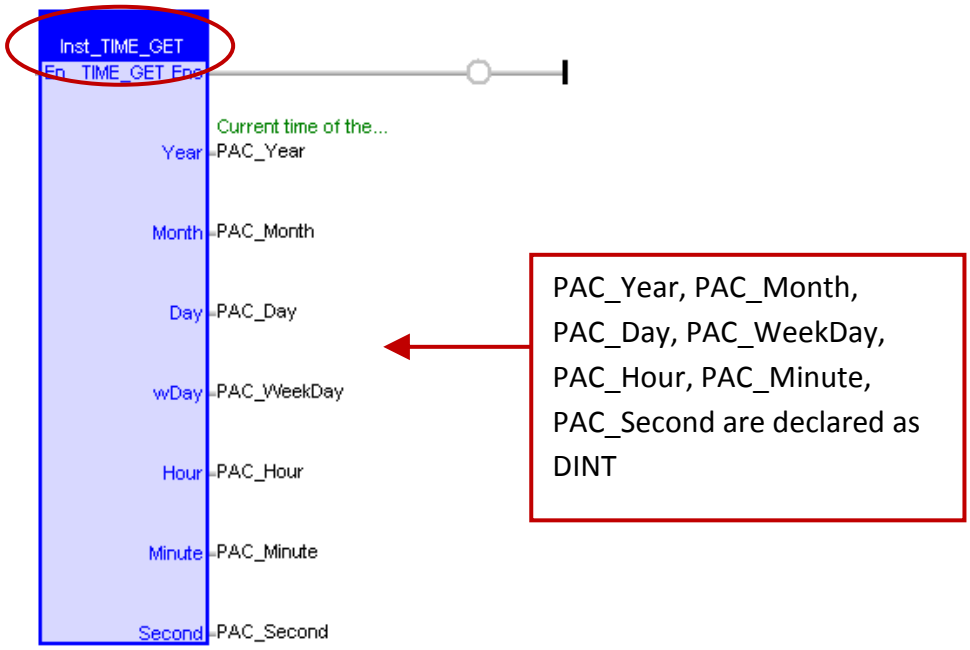
Example 4

(* To map one DINT_Val to one REAL_Val in the "Little Endian" sequence. *)

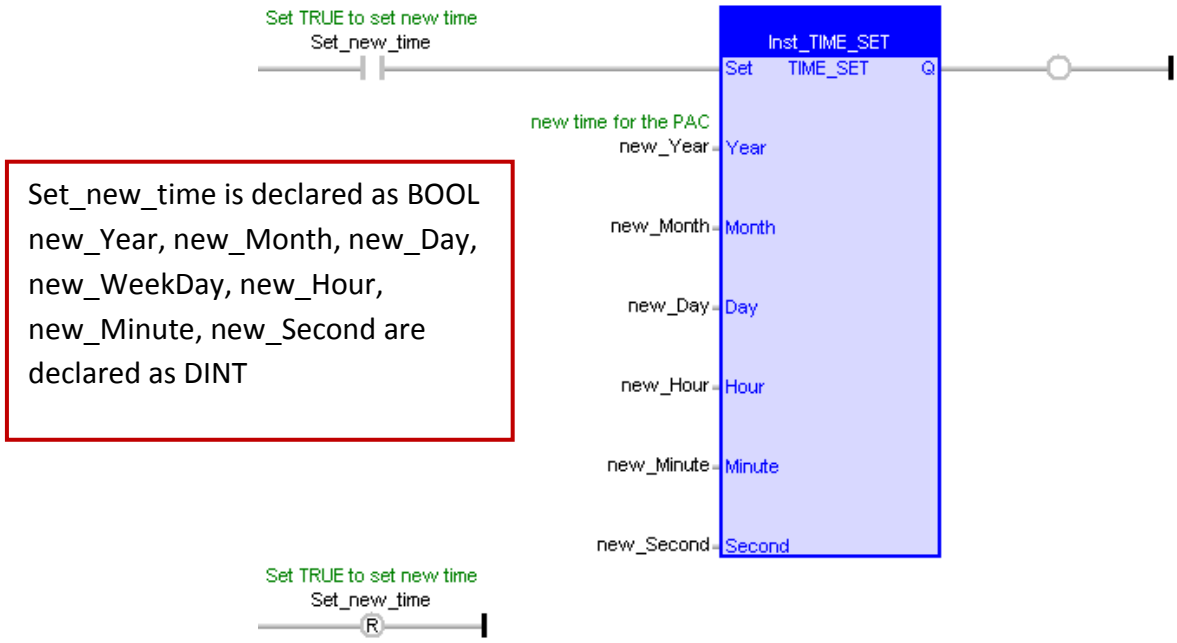
```
TMP_DINT := SerializeOut (buf, DINT_Val, 0, FALSE) ;
TMP_DINT := SerializeIn (buf, REAL_Val, 0, FALSE) ;
```

10.8 Get/Set the PAC Time

If you want to get the current time of a Win-GRAF PAC, you can use a "TIME_GET" Function Block. (Refer the Section 2.2.1)



If want to adjust the Win-GRAF PAC time, you can use "TIME_SET" Function Block. (Refer the Section 2.3.6) First, fill the new time to the variables of "new_Year", "new_Month", "new_Day", "new_WeekDay", "new_Hour", "new_Minute" and "new_Second", then set the "Set_new_time" to "TRUE" one time.



Chapter 11 Commonly Used Tools and Useful Tips

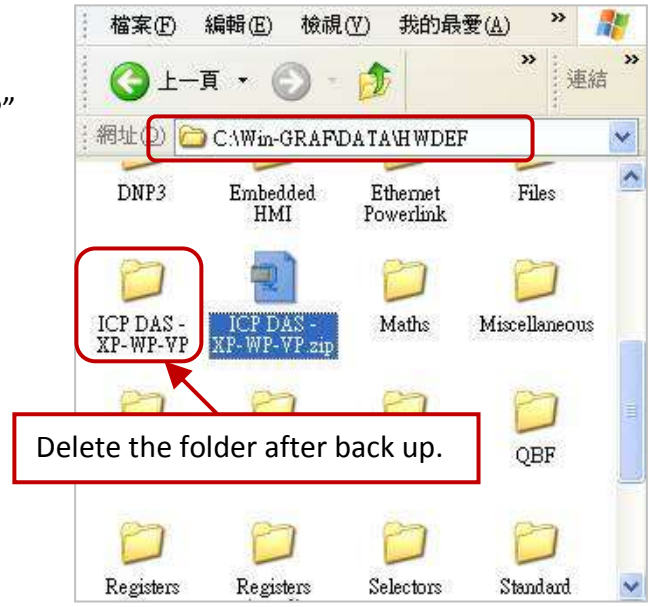
11.1 Upgrade Win-GRAF Libraries

Users can download the latest version of Win-GRAF libraries (Win-GRAF Lib, e.g., "win-graf-lib-x.xx.zip") from http://www.icpdas.com/root/product/solutions/softplc_based_on_pac/win-graf/download/win-graf-driver.html. The Win-GRAF Libraries (Including Function, Function Block and I/O Board definitions) are saved in the folder of "ICP DAS - XP-WP-VP" under the directory of "C:\Win-GRAF\DATA\HWDEF". In some situations, you need to upgrade the Win-GRAF Libraries to the new version for supporting more Functions or new I/O Board. Please follow the steps below:

1. First, close all Win-GRAF Workbench windows.

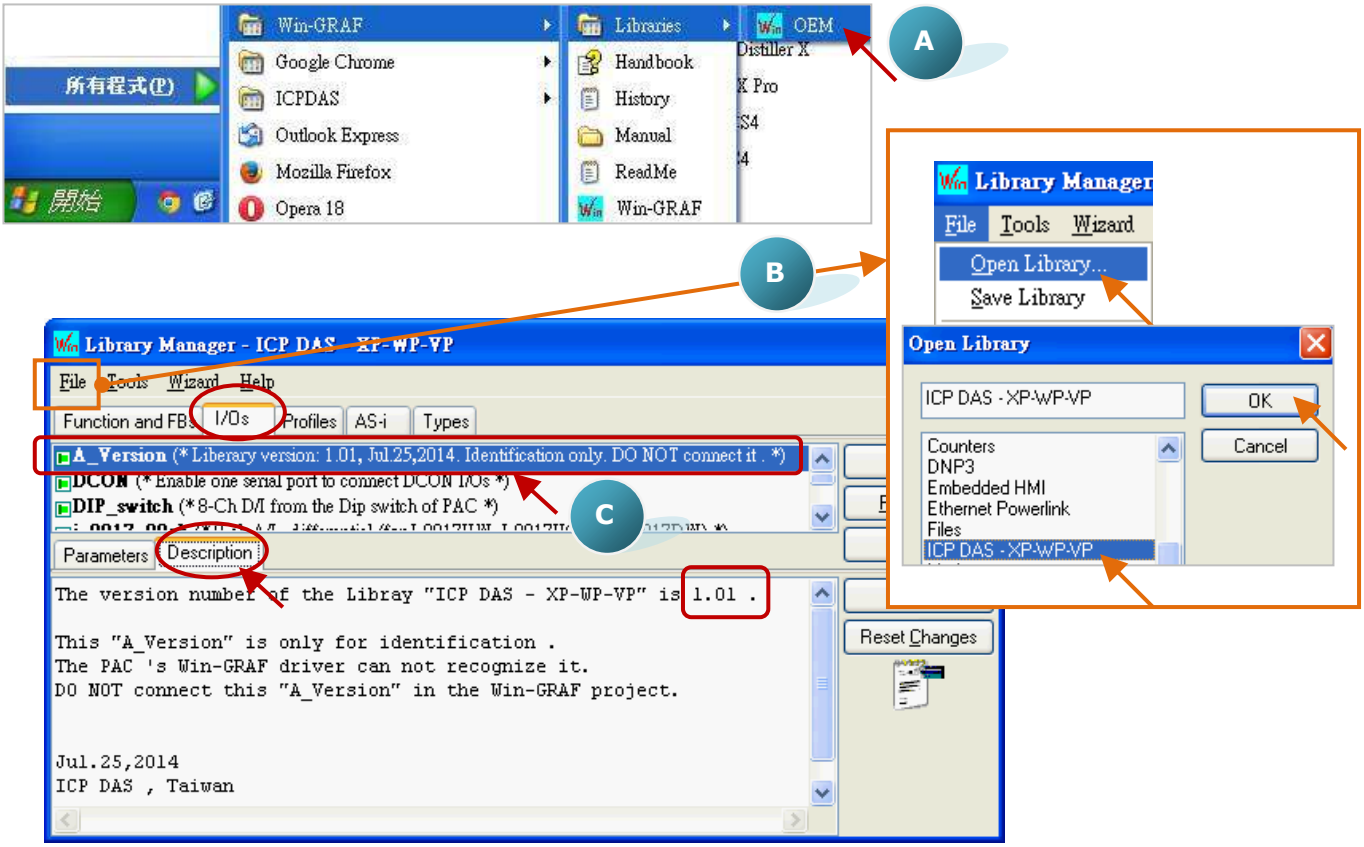
2. You can compress the original "ICP DAS - XP-WP-VP" folder and back it up to the other directory (e.g., D:\temp\xxx.zip), and then delete the folder.

3. Copy the new "ICP DAS - XP-WP-VP" folder into the directory "**C:\Win-GRAF\DATA\HWDEF**", and execute the Win-GRAF Workbench again.



Note:

If you want to know the version number of the Win-GRAF library. As the figure below, open the "Library Manager" and click "A_Version" in the "I/Os" tab, and then click "Description" to see the version number (e.g., "1.01").

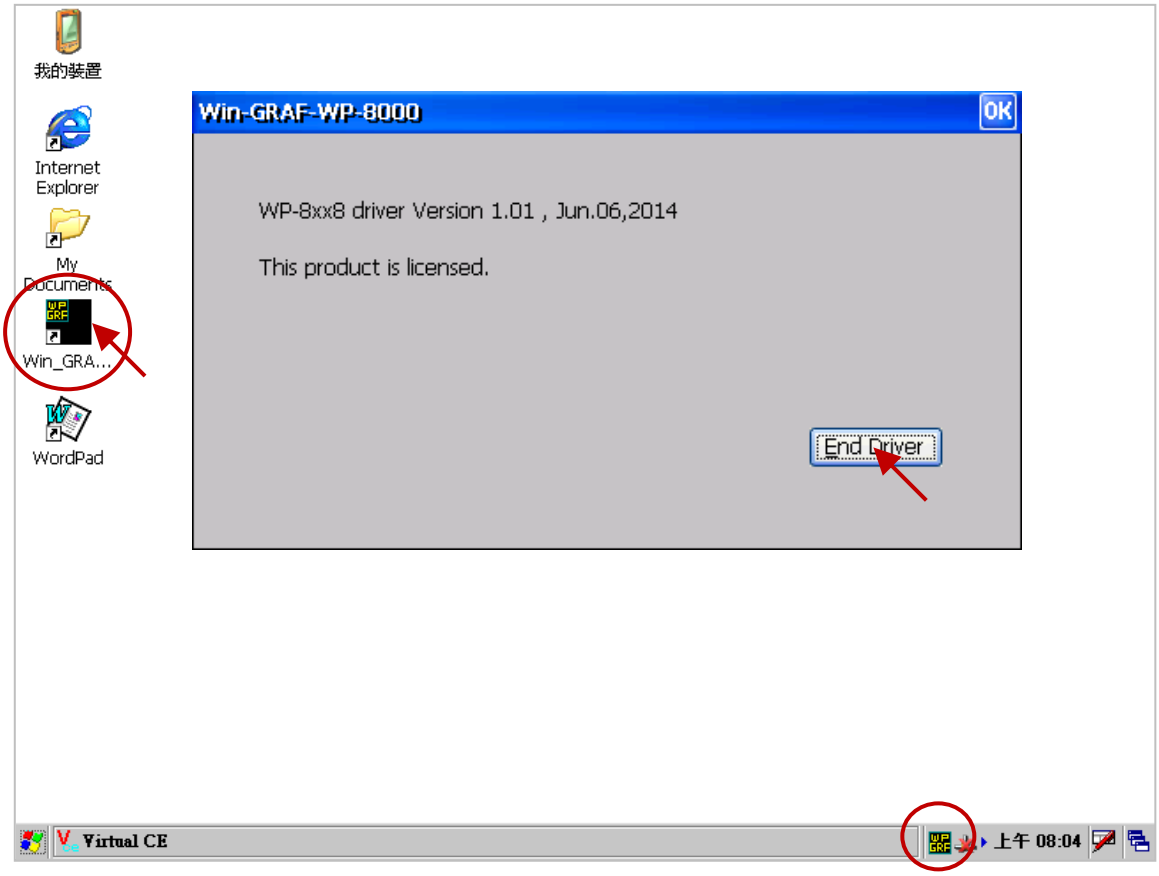


11.2 Upgrade Win-GRAF Driver

For updating add-on functions, I/O boards or other purposes, ICP DAS will release a new version of Win-GRAF drivers in the future. Users can get the latest driver on the website (http://www.icpdas.com/root/product/solutions/softplc_based_on_pac/win-graf/download/win-graf-driver.html), and follow the steps below to upgrade the new driver into the PAC.

Note: The Win-GRAF Driver of the XPAC (XP-8xx8-CE6), WinPAC (WP-8xx8, WP-8xx8-CE7, WP-5xx8-CE7) and ViewPAC (VP-x2x8-CE7) will be placed in the directory "\System_Disk\Win-GRAF\" inside the PAC.

1. On the desktop of a PAC (use WP-8xx8 in this example), double click on the "Win-GRAF_WP_8000" icon and then click "End Driver" to stop the currently running driver.



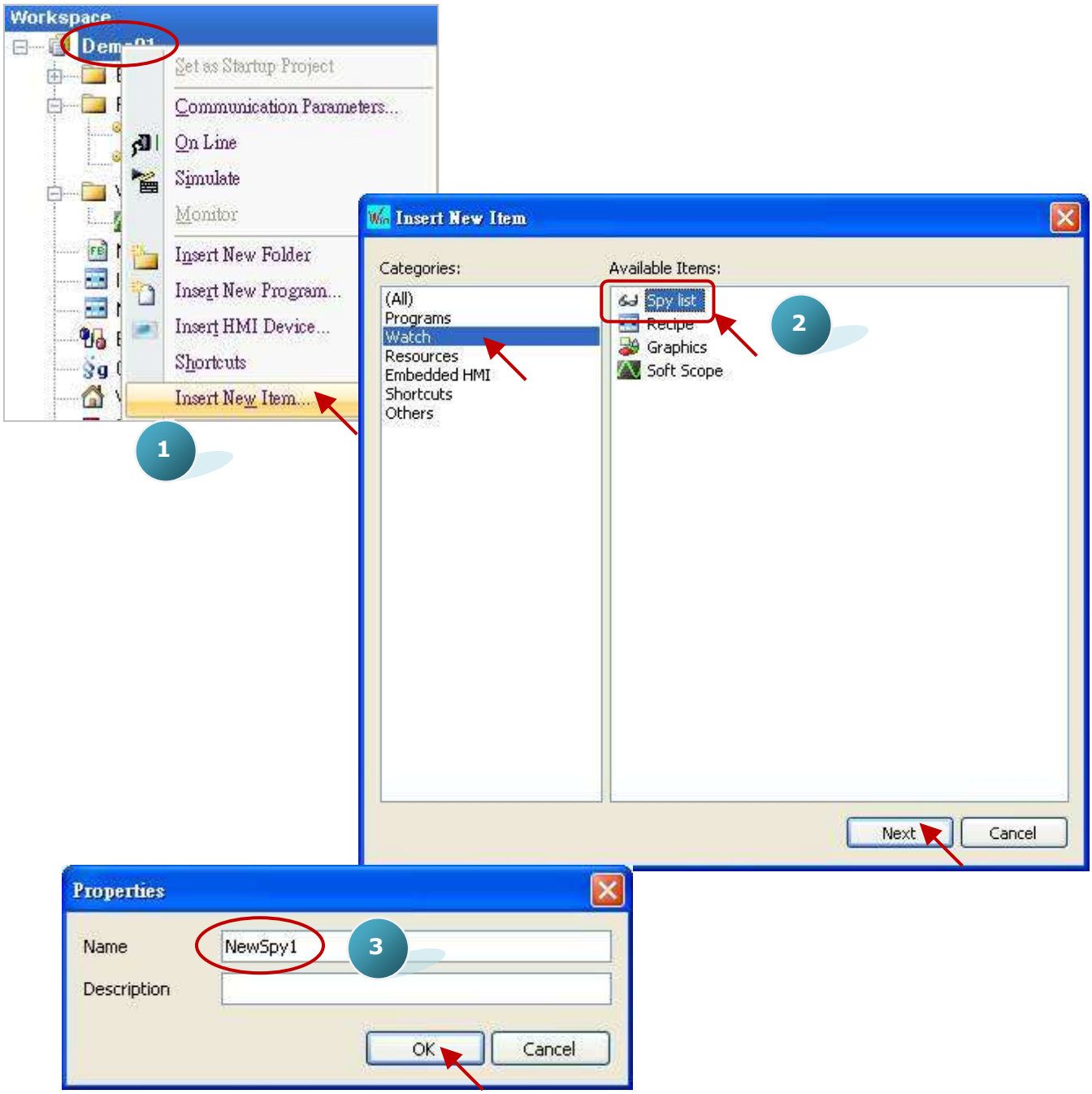
2. On the PC, copy the new driver into the PAC's directory “\Temp\” by using FTP method.
3. On the PAC, copy the new driver from “\Temp\” into “\the System_Disk\Win-GRAF\” directory to replace the old one, and then reboot the PAC.

11.3 Spy List

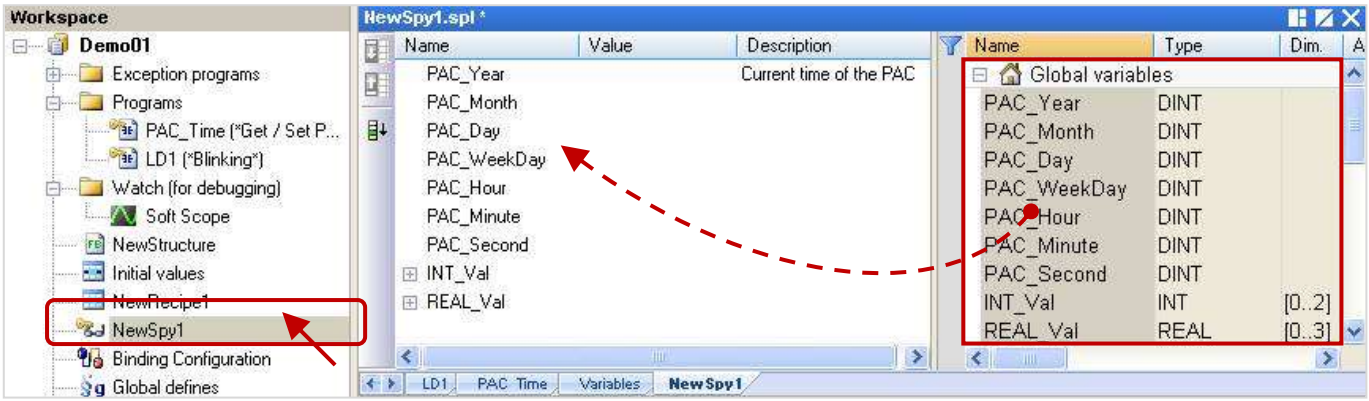
When a program is running, the Spy List lets users quickly know the variable's value or status. Sometimes, a program may declare hundreds or thousands of variables, users do not need to look for them, just simply switch to the pre-created Spy List window to see the wanted information.

Steps:

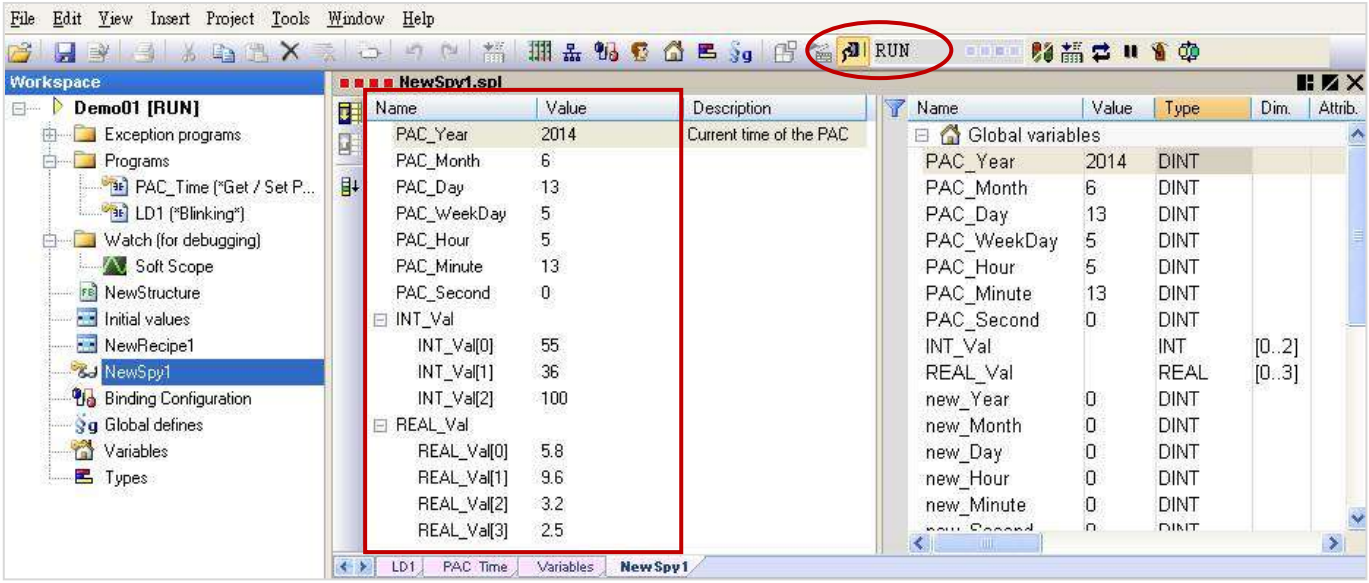
1. Right-click on the project name (e.g., "Demo01") and select "Insert New Item".
2. Select "Spy List" of the "Watch", then click "Next" to the next step.
3. Then, key in a list name (e.g., "NewSpy1") and press "OK".



4. Double click "NewSpy1" on the left side to open the setting window and drag the variables you want to put into the window.



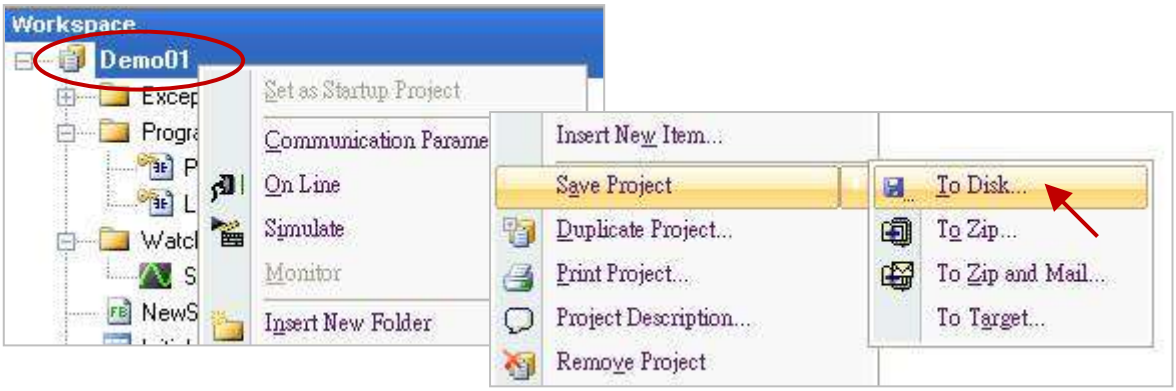
5. When the Win-GRAF and the PAC are connected, the "NewSpy1" window will show the variables information clearly.



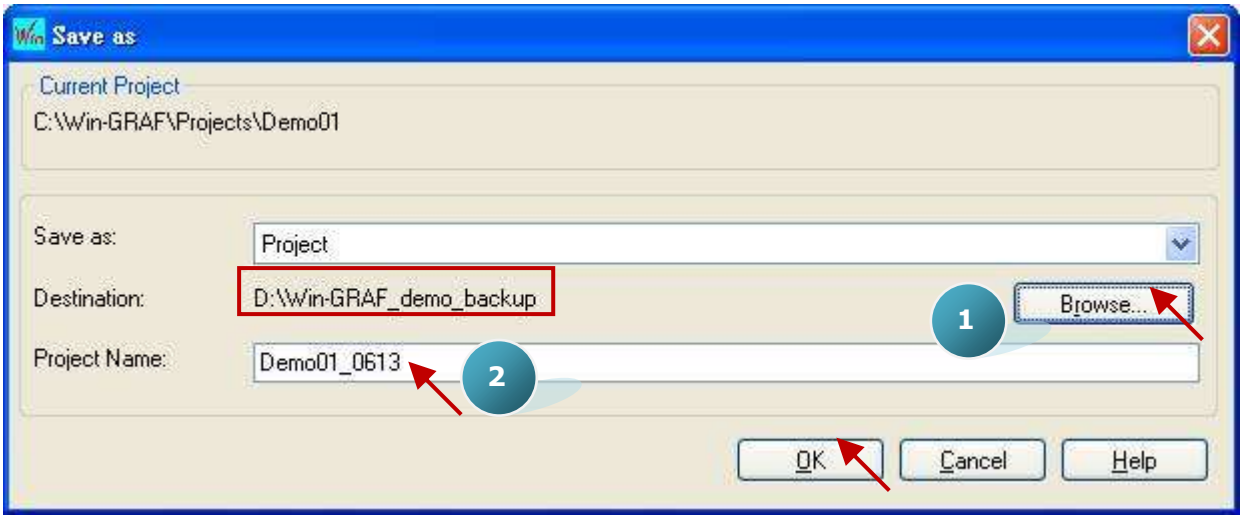
11.4 Backup/Restore a Win-GRAF Project

Back up A Win-GRAF Project:

1. Mouse right-click on the project name (e.g., "Demo01") and select "Save Project" and then "To Disk".

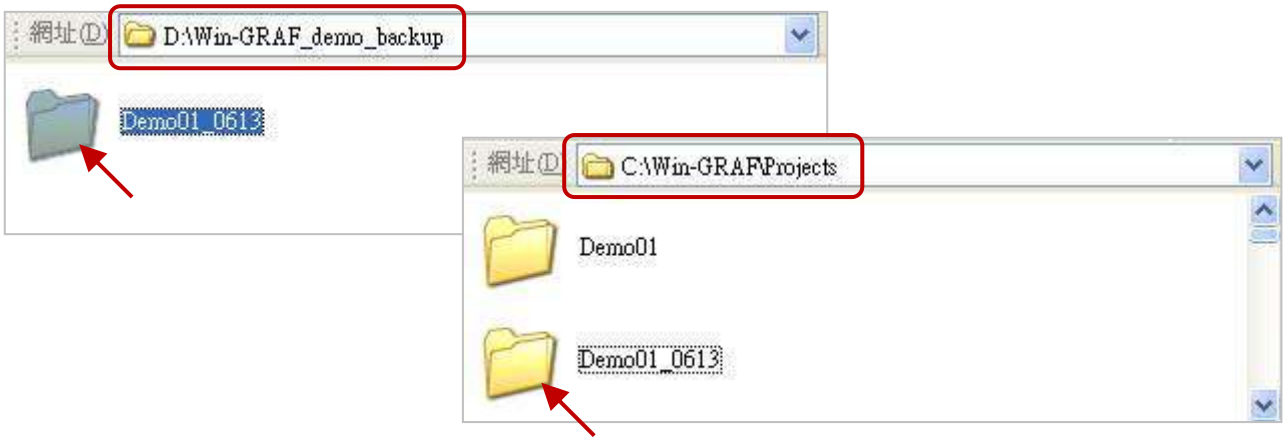


2. Click on the "Browser" button to assign a directory you want to save the project (e.g., D:\Win-GRAF_demo_backup), fill in the project name (e.g., "Demo01_0613"), and then click "OK" to back up the project.

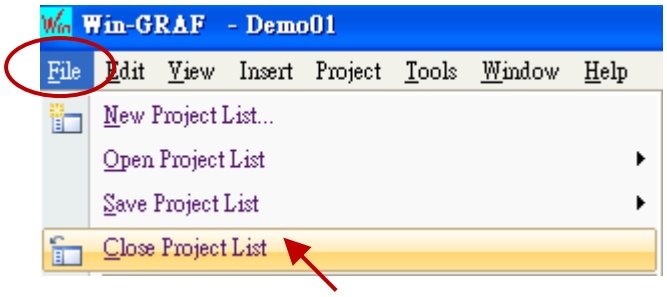


Restore A Win-GRAF Project:

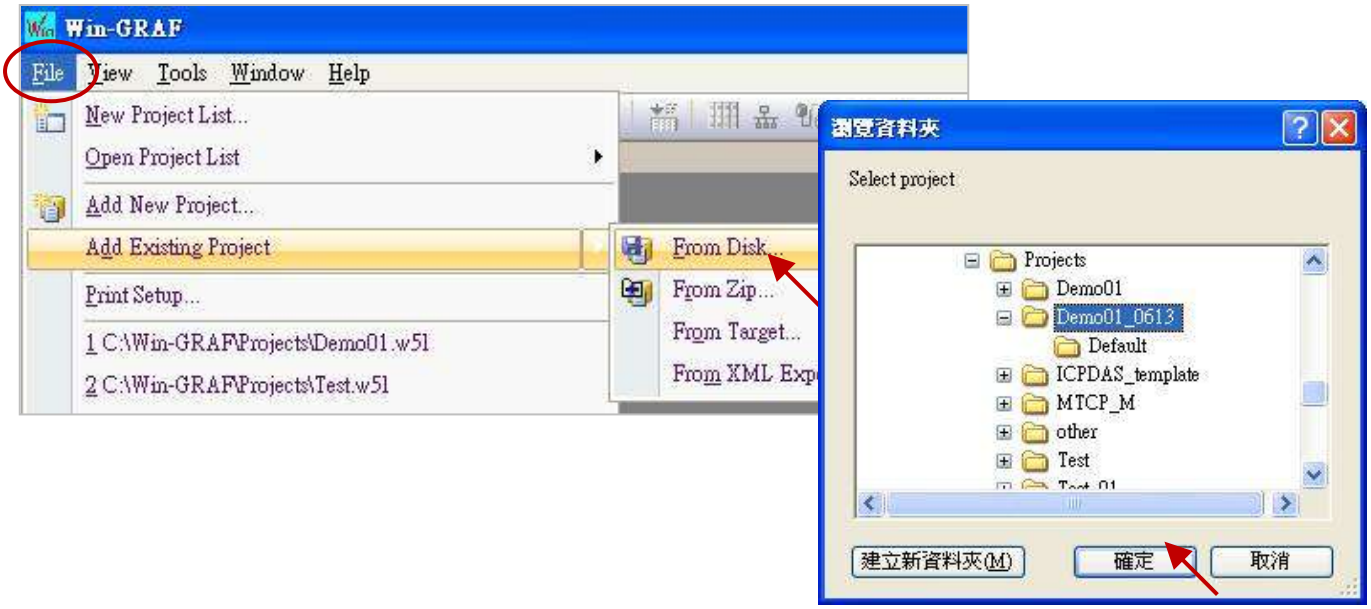
1. Copy the previously backed up the project folder (e.g., "Demo01_0613") into "C:\Win-GRAF\Projects".

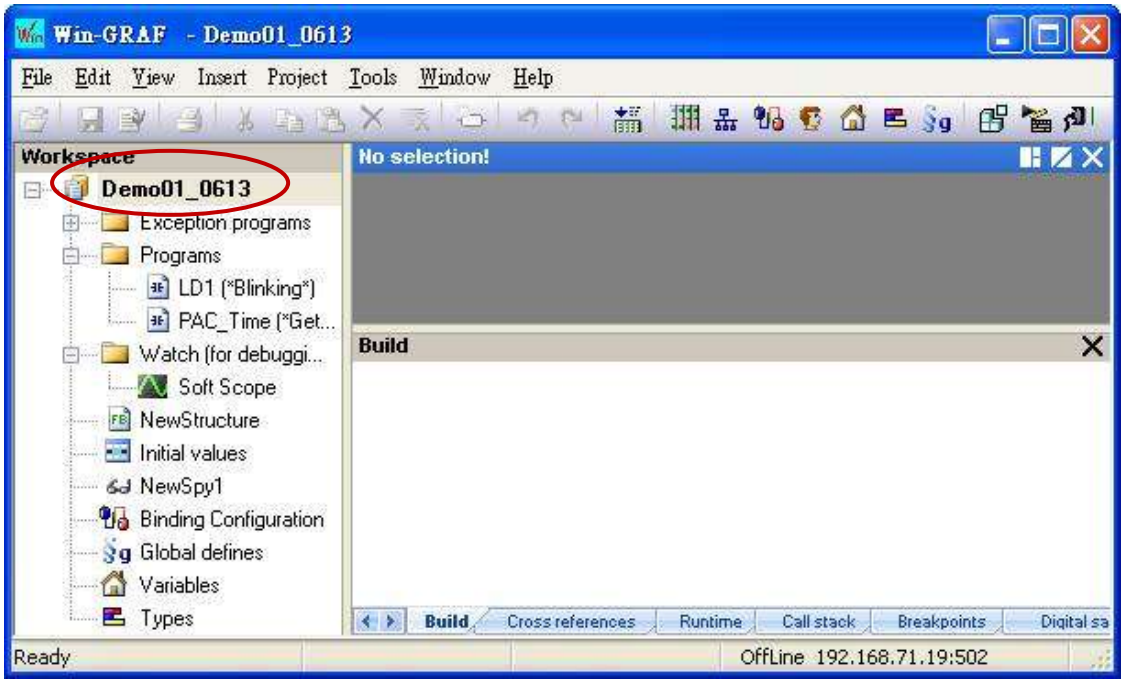


2. Click the menu bar "File" > "Close Project List" to close all opened project windows.



3. Click the menu bar "File" > "Add Existing Project" > "From Disk", select the project you want (e.g., "Demo01_0613") in the "C:\Win-GRAF\" directory, and then click "OK" to restore the project.





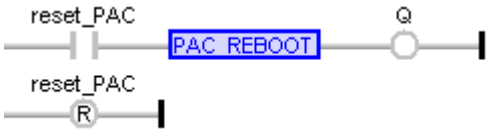
11.5 Software Reboot a PAC

Based on some cases, users may want to reboot the PAC in a software way. The Win-GRAF provides a Function "PAC_Reboot" for users to restart the PAC.

Note: Please DO NOT call this Function in every PAC Cycle, or the PAC will reboot all the time.

Safety Coding:

```
(*  "reset_PAC" is declared as BOOL and has initial "FALSE"
    "TMP_BOOL" is declared as BOOL *)
 (* ONLY when "reset_PAC" is set to "TRUE", the PAC will reboot  *)
    if  reset_PAC  then
      reset_PAC := FALSE ;
      TMP_BOOL  :=  PAC_Reboot( ) ;
    end_if ;
```



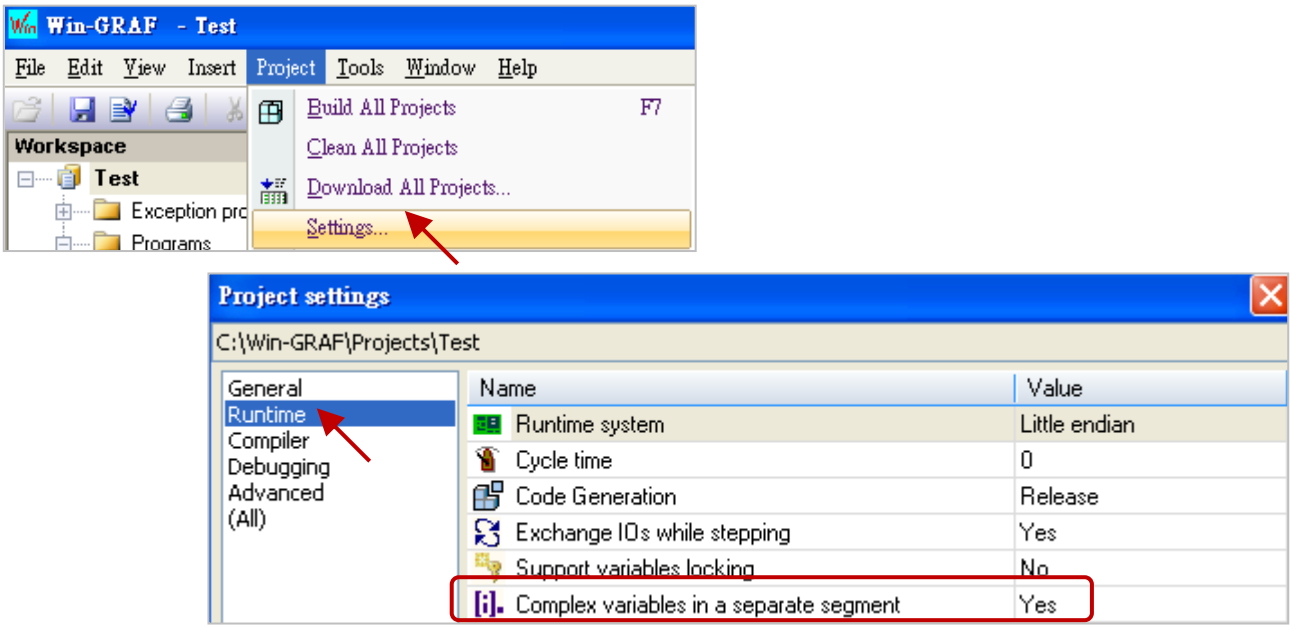
Dangerous Coding:

```
(*  "TMP_BOOL" is declared as BOOL *)
(* Dangerous ! This coding method will let the PAC reboot always and cannot stop. *)
    TMP_BOOL  :=  PAC_Reboot( ) ;
```

If a mistake to reboot the PAC always, turns the rotary switch of the Win-GRAF PAC to "1" and reset it. Then it will boot up in safe mode. Then you may rename the Win-GRAF application code on the PAC to an invalid name. Then when the rotary switch is turned back to "0" and reboot, it will boot up normally (No application). The Win-GRAF application code in the XP-8xx8-CE6, WP-8xx8, WP-8xx8-CE7, WP-5xx8-CE7, and VP-x2x8-CE7 is "\System_Disk\Win-GRAF\t5.cod".

11.6 Using ST Syntax in LD and FBD

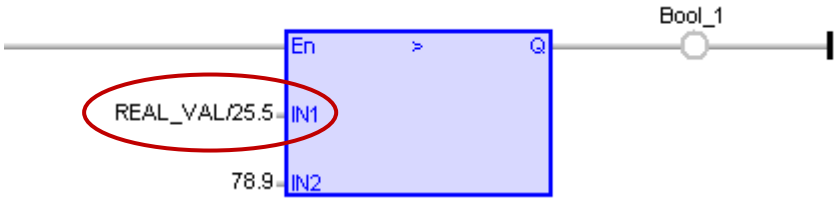
The Win-GRAF Workbench allows users using simple ST syntax in Ladder (LD) and Function Block Diagram (FBD) to facilitate programming. Before use, go to the menu bar "Project"> "Settings" > "Runtime", and set the "Complex variables in a separate segment" to "Yes" to enable this function.



Example:

<u>LD Program:</u>

Using division (REAL_VAL/25.5).



<u>FBD Program:</u>

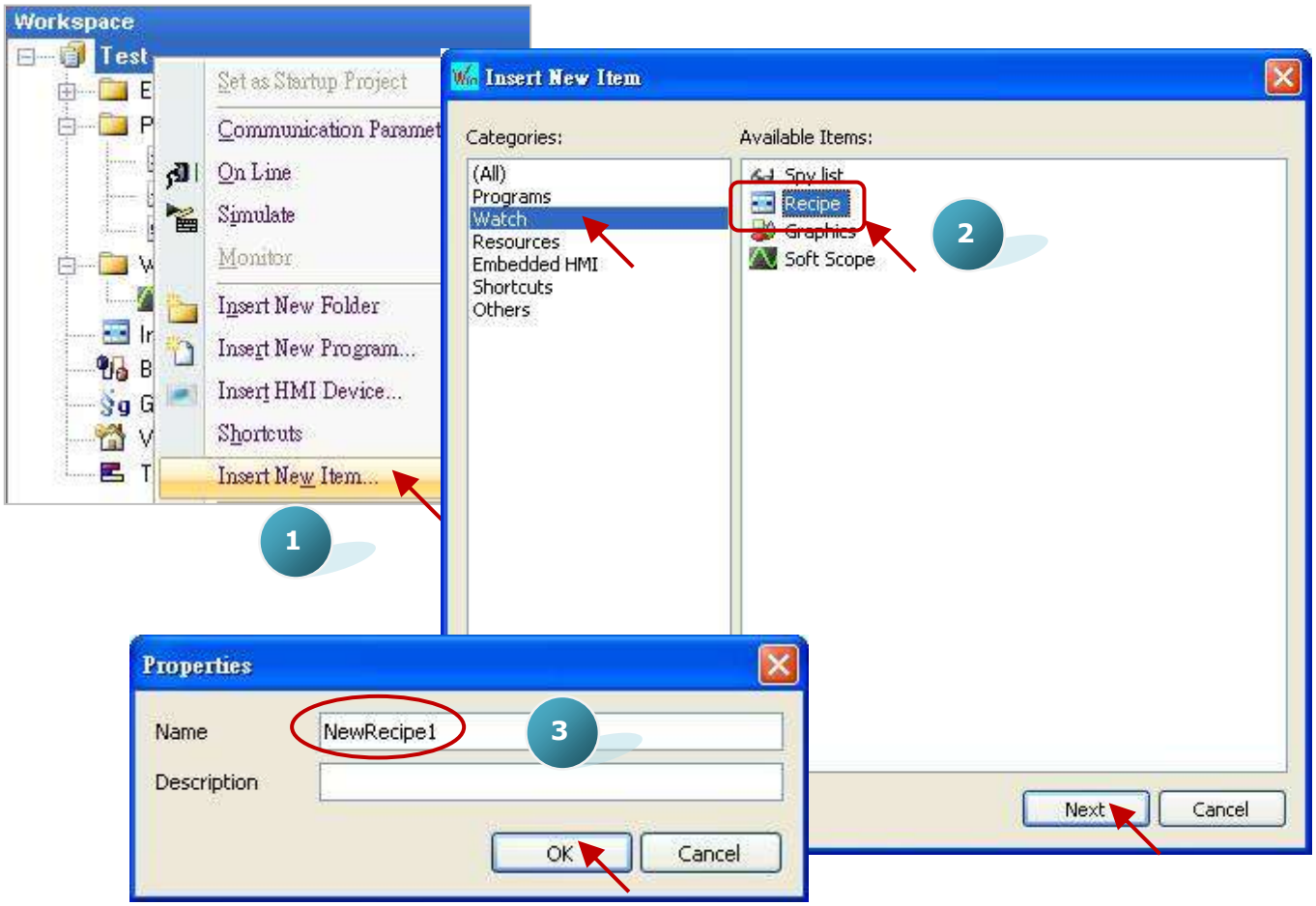
Call a function "ANY_TO_BYTE()" to convert the type from "SINT" to "BYTE".

11.7 Apply a Recipe on the PAC

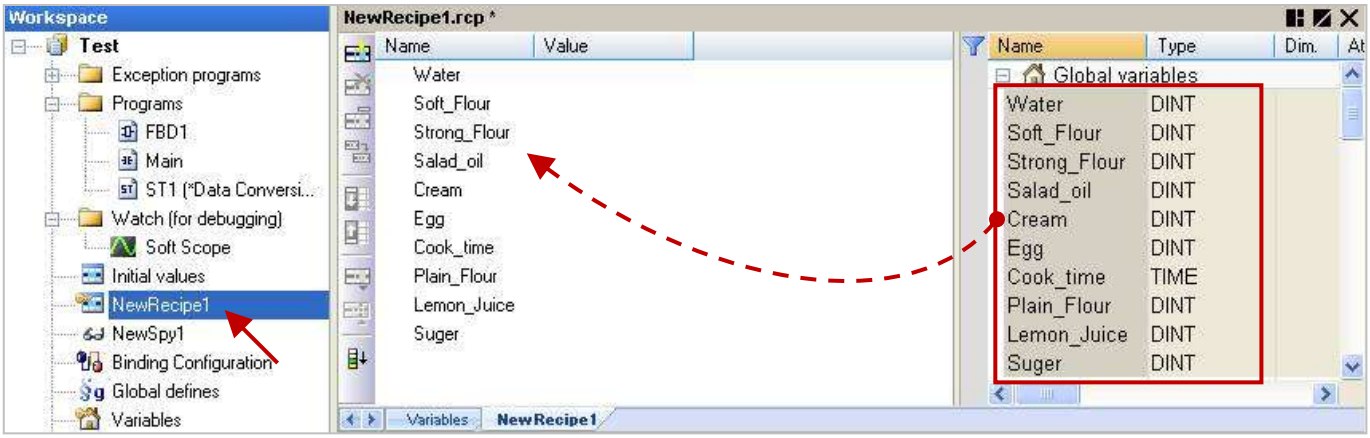
Some applications use the pre-defined Recipe and Value for processing different products, and this Recipe can be mapped to a combination of variables within a Win-GRAF PAC. When one day want to change the PAC process to produce a different product, you can use the Win-GRAF Workbench to connect with the PAC and select a new Recipe you want to replace, and then apply it to the PAC.

Steps:

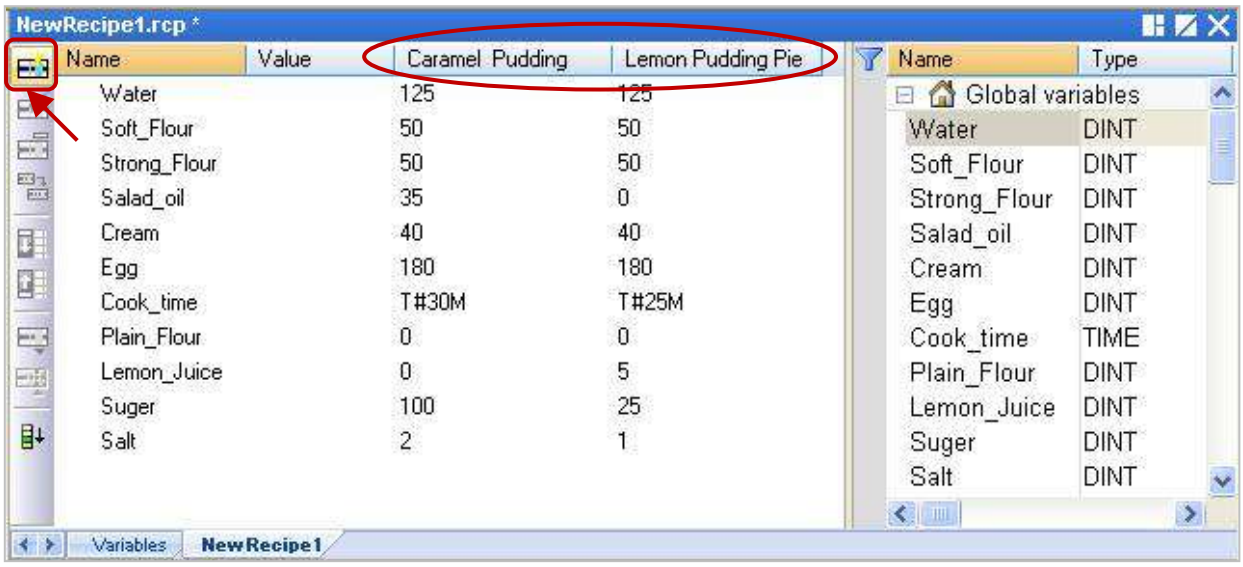
1. Mouse right-click on the project name (e.g., "Test") and select "Insert New Item"; then click on "Watch" > "Recipe" and "Next" button, fill in the Recipe name (e.g., "NewRecipe1") and then click "OK".



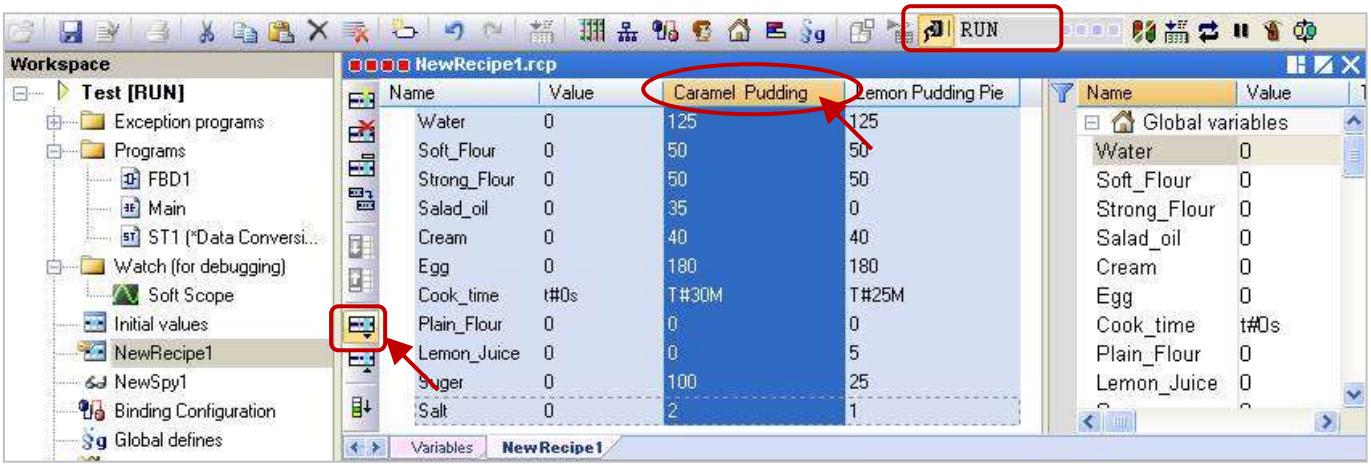
2. Double click "NewRecipe1" on the left side to open the setting window and drag the variables you want to put into the window.

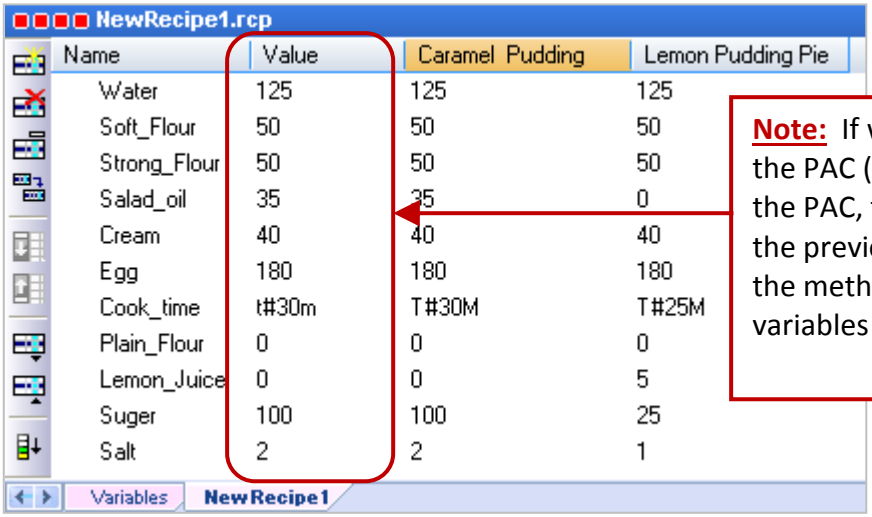


3. Click on the "Insert Column" icon to add the new Recipe, and then fill in the suitable Values.



4. Click on the "On Line" button to connect the PAC. At first the Values are all "0", please select the product column and then click the icon "Send Recipe" to apply this Recipe into the PAC.

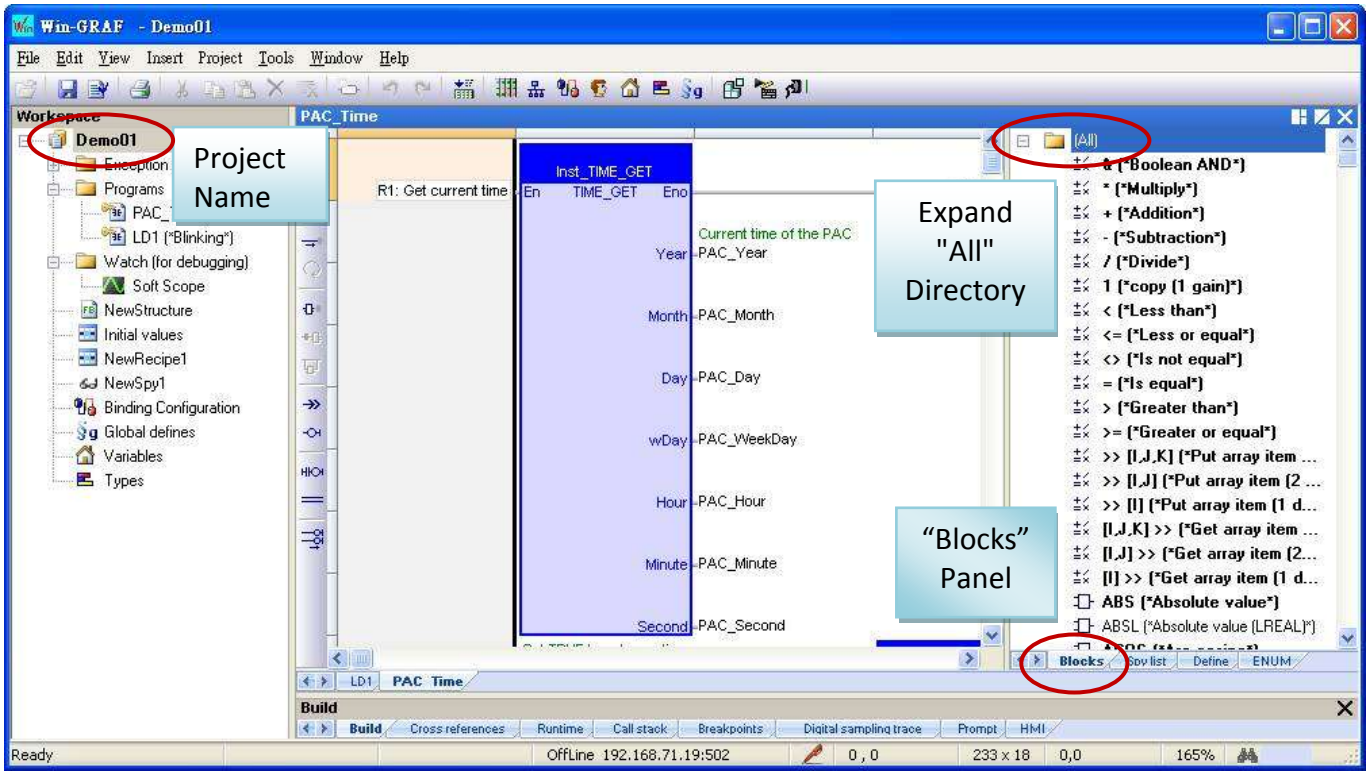




Note: If want to save the Values on the PAC (e.g. Power off and restart the PAC, the Recipe can still retain the previous Values), please refer the method of using the retain variables in the Section 6.1.

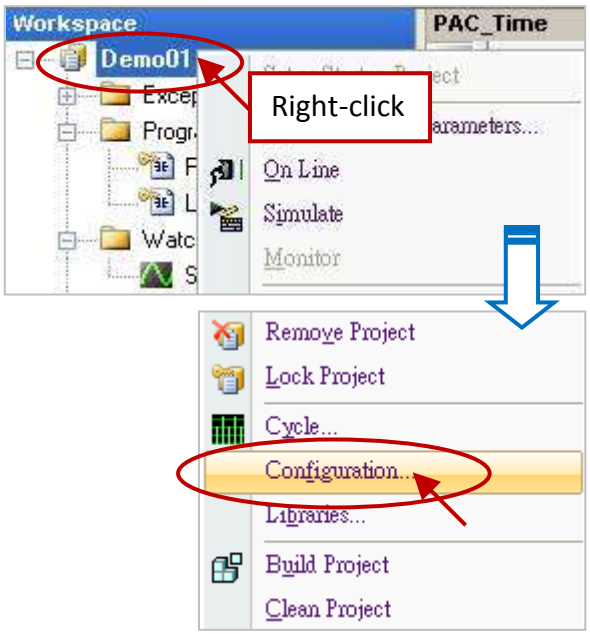
11.8 Get the Functions and Function Blocks that Supported by the PAC

In the Win-GRAF Workbench window, user can expand the "All" directory in the "Blocks" panel to see quite a lot of Functions and Function Blocks, however, some are not supported in the Win-GRAF PAC. The following will show how to quickly get the Functions or Function Blocks that supported by the PAC.

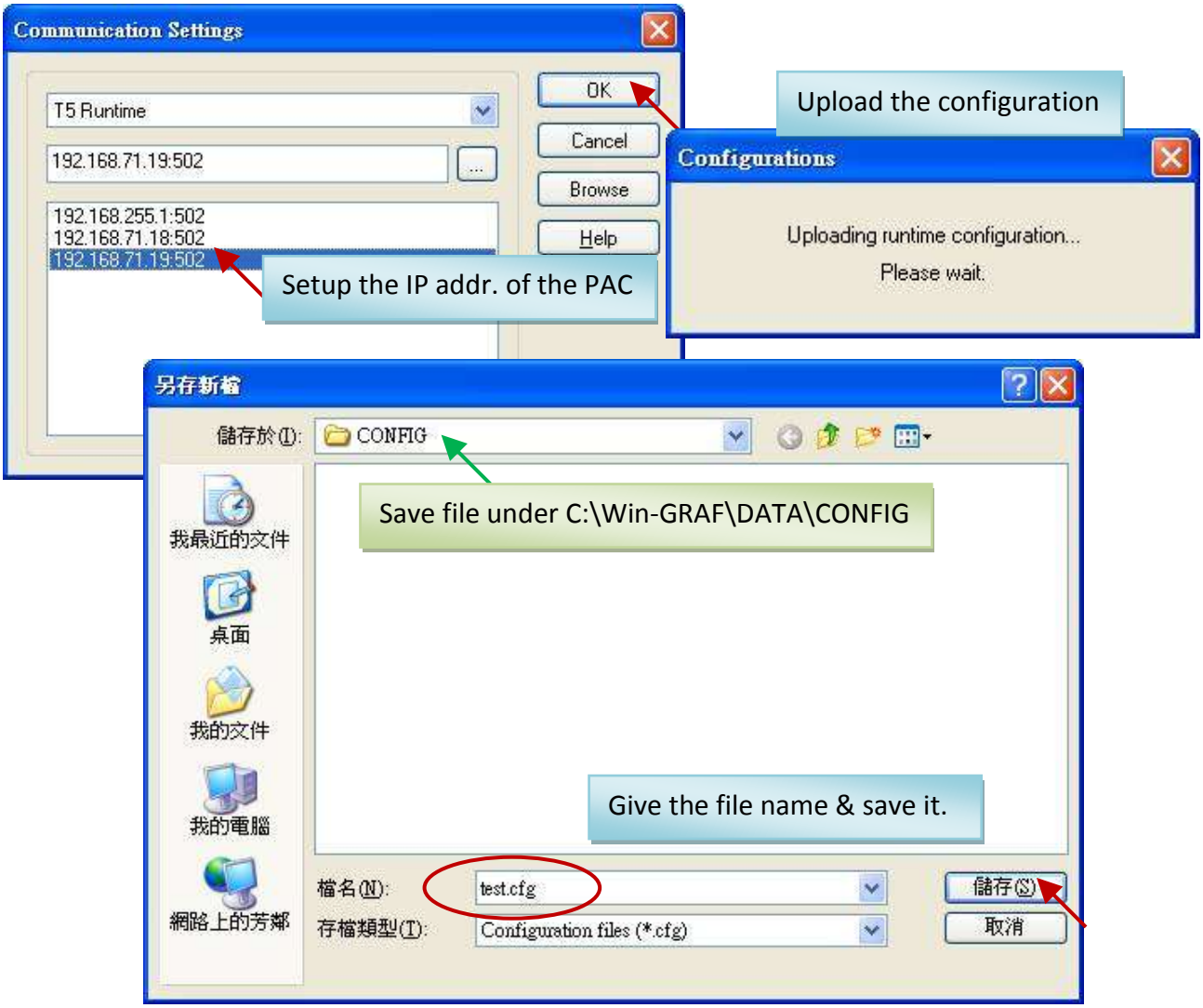


Setting Steps:

1. Make sure the PAC is powered on and connected with a PC via an Ethernet cable.
2. In the Win-GRAF Workbench, right-click on the project name (e.g., "Demo01") and then select "Configuration", and click on the "Upload" button in the "Select" tab to open the setting window.

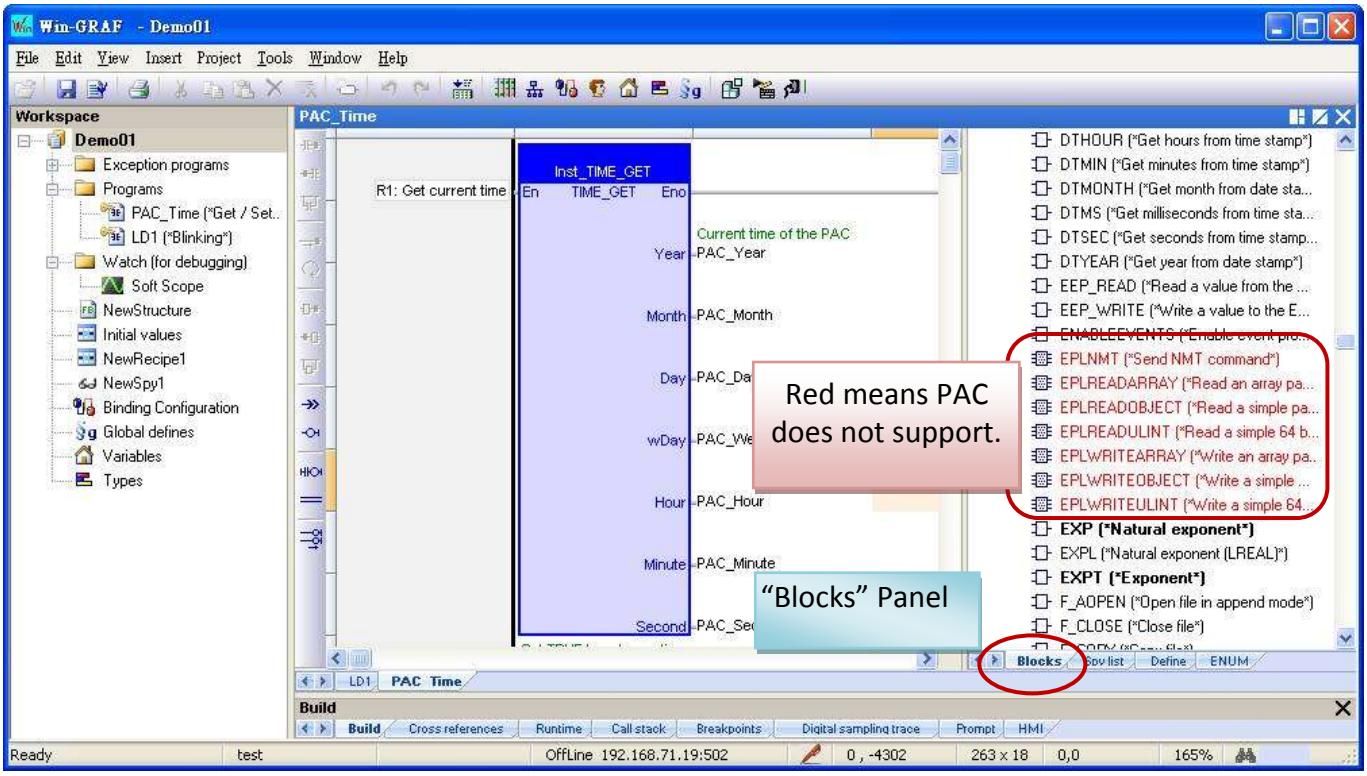


3. Configure/Select the IP address of the PAC and click on "OK" button, the PAC will upload the configuration file. Next, key in the file name (e.g., "test.cfg") and click "Save" to save the configuration file.



4. Back to the "Configurations" window, the configuration file (test) will show in the list, and then click "OK" to leave this window.

5. In the "Blocks" panel, the red Functions and Function Blocks are not supported by this PAC.

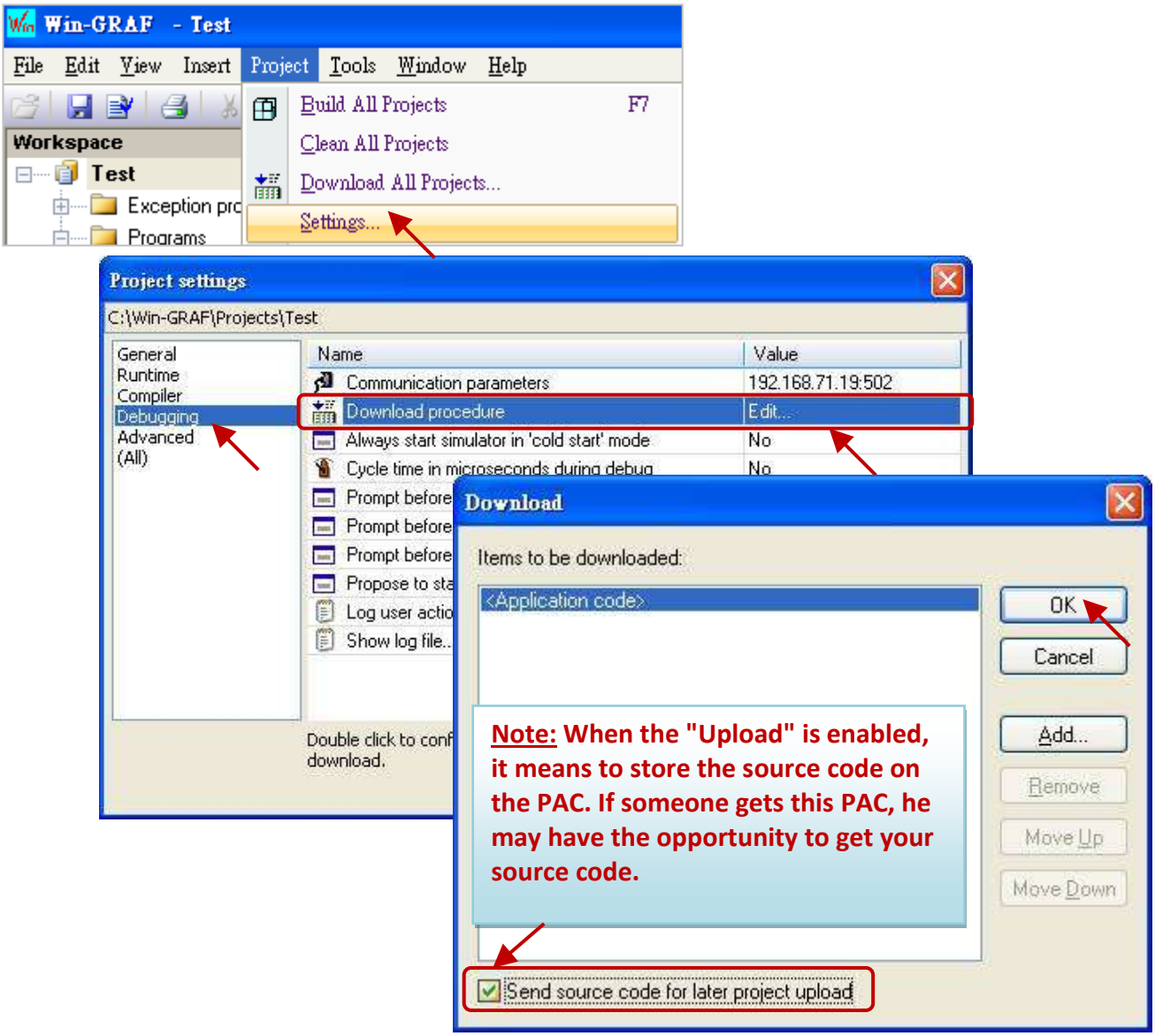


11.9 Upload the Win-GRAF Source Code

For some applications, users need to get the source code of the Win-GRAF project from the PAC to the PC, this is called "Upload". This function can prevent the project source code from missing or incomplete handover from the previous worker, you can still get the project source code inside the PAC.

Enable/Download The Project Source Code:

1. Click on the menu bar "Project" > "Settings" to open the setting window.
2. Double click on the "Download procedure" of the "Debugging" and select "Send source code for later project upload", and then click "OK ".

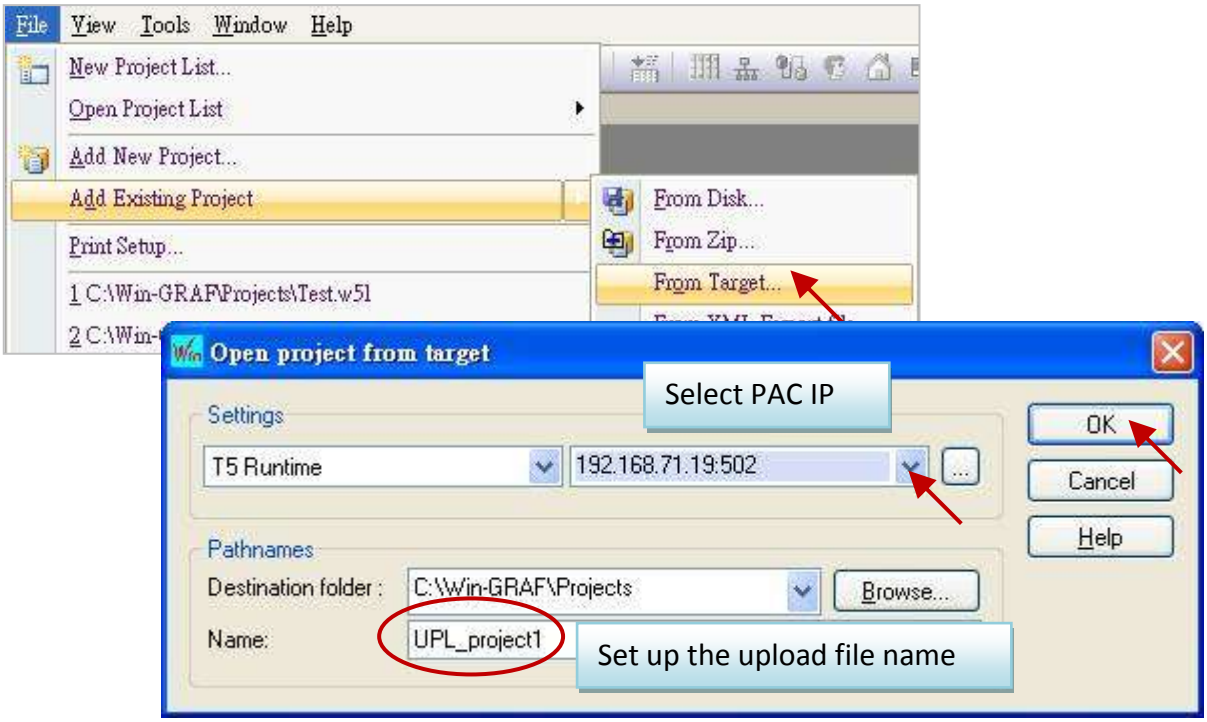


3. Click on the menu bar "Project" > "Build All Projects" to compile the program, and then click on the tool icon  to connect the PAC, and next, download the current project to the PAC. (Refer the Section 2.3.5 for the detail steps). After downloading, the source code will be stored in the file of the directory "\Systen_Disk\Win-GRAF\t5.upl" of the PAC. This file will be larger when the project increases. If the project becomes very large and complex, the file size may reach several hundreds K Bytes or even more than 1 MB.

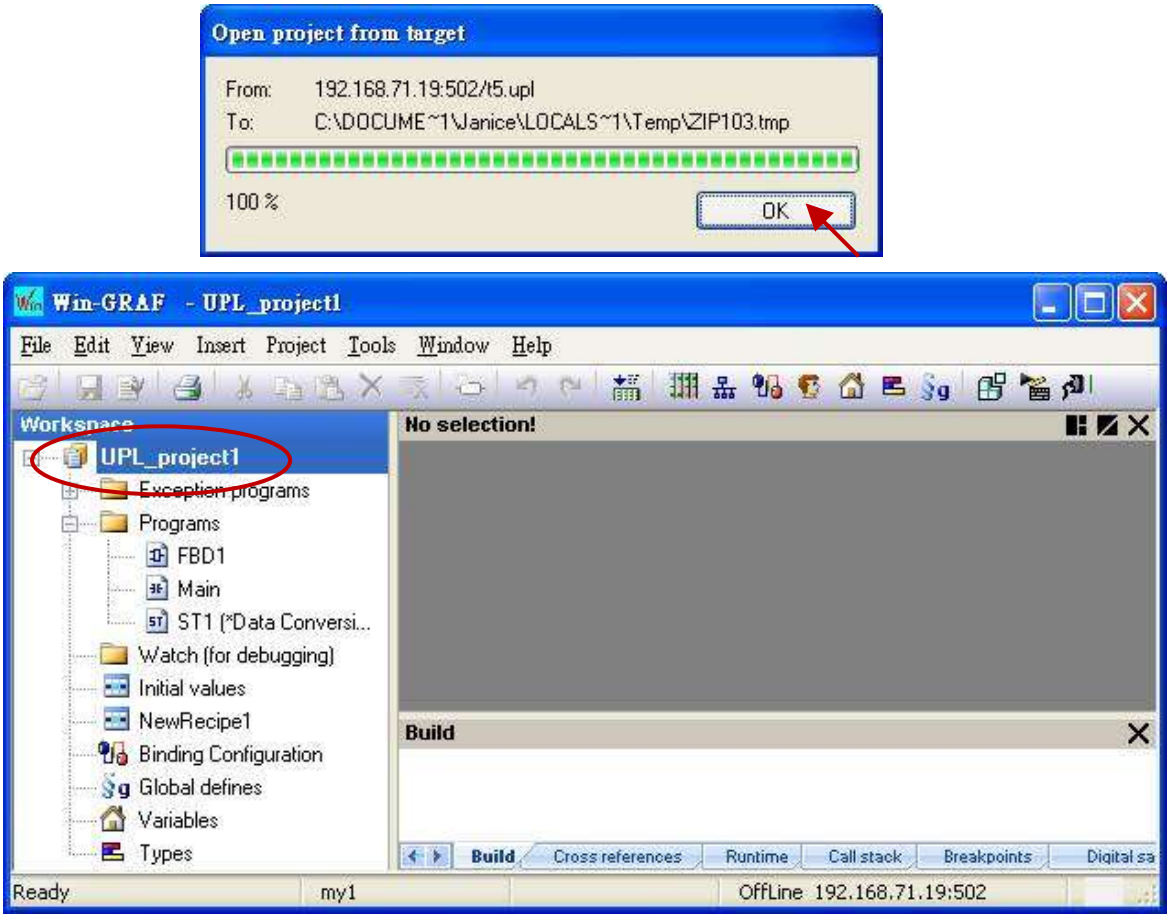
Upload the Porject Source Code:

Please close all opened Win-GRAF windows (Click on the menu bar "File" > "Close Project List").

4. Click on the menu bar "File" > "Add Existing Project" > "From Target", then select the PAC's IP address and set up the upload file name (e.g., "UPL_project1"), and then click "OK" to upload the file.



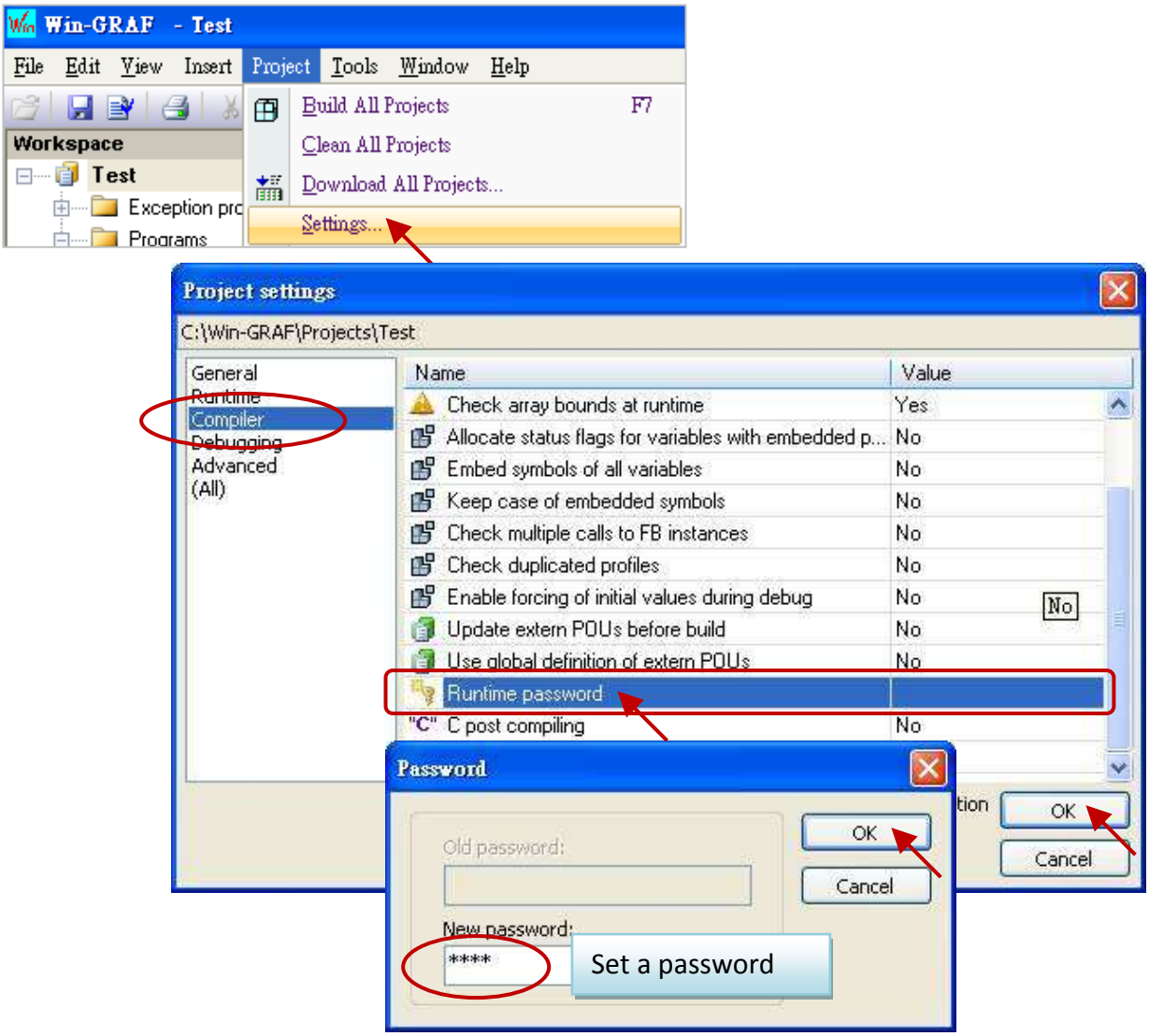
5. After uploading, click on "OK" button, and then the Win-GRAF will open the project automatically.



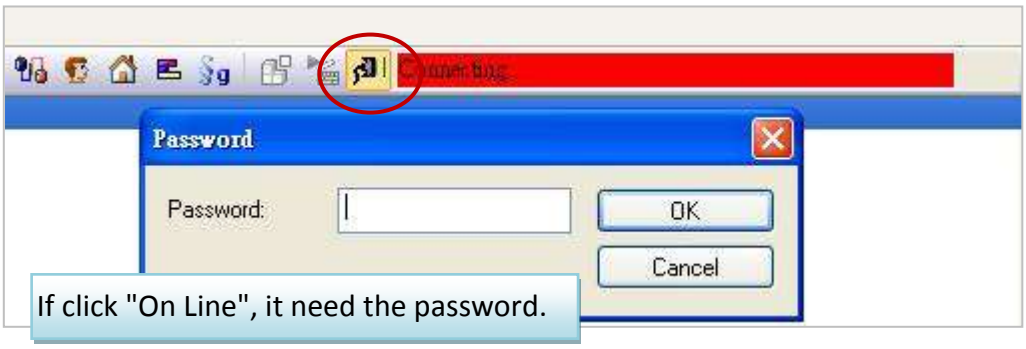
11.10 Set Up the PAC Password

In order to avoid the important program running on the PAC is changed or stopped by an unfriendly connecting PC, you can set up a password for the PAC to prevent unauthorized operation.

1. Click on the menu bar "Project" > "Settings" to open the setting window.
2. Double click on the "Runtime password" of the "Compiler", set a password, and then click "OK".



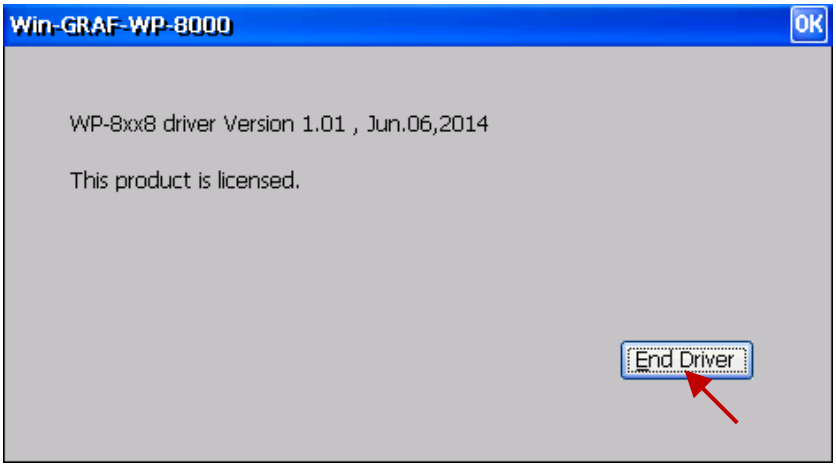
3. Click on the menu bar "Project" > "Build All Projects" to compile the program again, and then download the current project to the PAC (Refer the Section 2.3.5 for detail steps.). When the next time to click the "On Line" icon for connection, it will require the password.



Note: After enabling the password, please remember your password, or you will not connect the PAC.

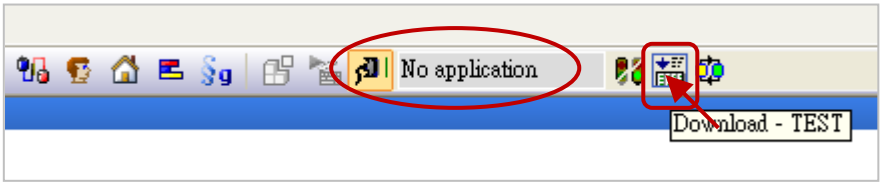
The Only Solution:

1. Connect the PAC with a USB mouse and screen.
2. On the PAC, execute the Win-GRAF Driver and then click on "End Driver" button. (Refer the Section 11.2).



3. Rename the file "t5.cod" in the directory "\System_Disk\win-graf" (e.g., "t5.cod1") or delete it. Then reboot the PAC.

Then, it will become "No application" on the PAC. Now, you can connect and download the application from the Win-GRAF Workbench again.



11.11 Using Function Block in the ST Program

It is easy to use the Function in the ST program, just call the Function and assign the corresponding parameters. The example below will open COM3 at the beginning, and then send a String `Hello' from COM3 every 5 seconds.

```
(* Declare  "INIT1" as BOOL and has initial value TRUE,
   Declare  "TMP_BOO" as BOOL,  "TMR1"  as TIME  *)

 IF    INIT1    THEN
   INIT1  :=  FALSE ;
      TMR1  :=  T#0s ;
   TSTART  (TMR1) ;
 END_IF;
 IF  COM_Status(3) = FALSE  THEN
   TMP_BOO := COM_open (3, `19200,N,8,1') ;
 END_IF ;
 IF    TMR1  >=  T#5s    THEN
   TMR1  :=  T#0s ;
    COM_send_str (3, `Hello: ') ;
 END_IF;
```

To use a Function Block in the ST, you must first declare an Instance Variable of the Function Block in the variable region, after that, the using steps are similar to the steps of using the Function, as follows:

The following code can unpack one Byte to become 8 BOOLs:

1. Declare "MyUnpack" variable as "UNPACK8" (FB Instance) and "IN" variable as "BYTE".

2. Edit an ST program.

```
MyUnpack(IN) ;
Q0  :=  MyUnpack.Q0 ;
Q1  :=  MyUnpack.Q1 ;
Q2  :=  MyUnpack.Q2 ;
Q3  :=  MyUnpack.Q3 ;
Q4  :=  MyUnpack.Q4 ;
Q5  :=  MyUnpack.Q5 ;
Q6  :=  MyUnpack.Q6 ;
Q7  :=  MyUnpack.Q7 ;
```

11.12 How to Protect Your Win-GRAF Program to Avoid Unauthorized Copied?

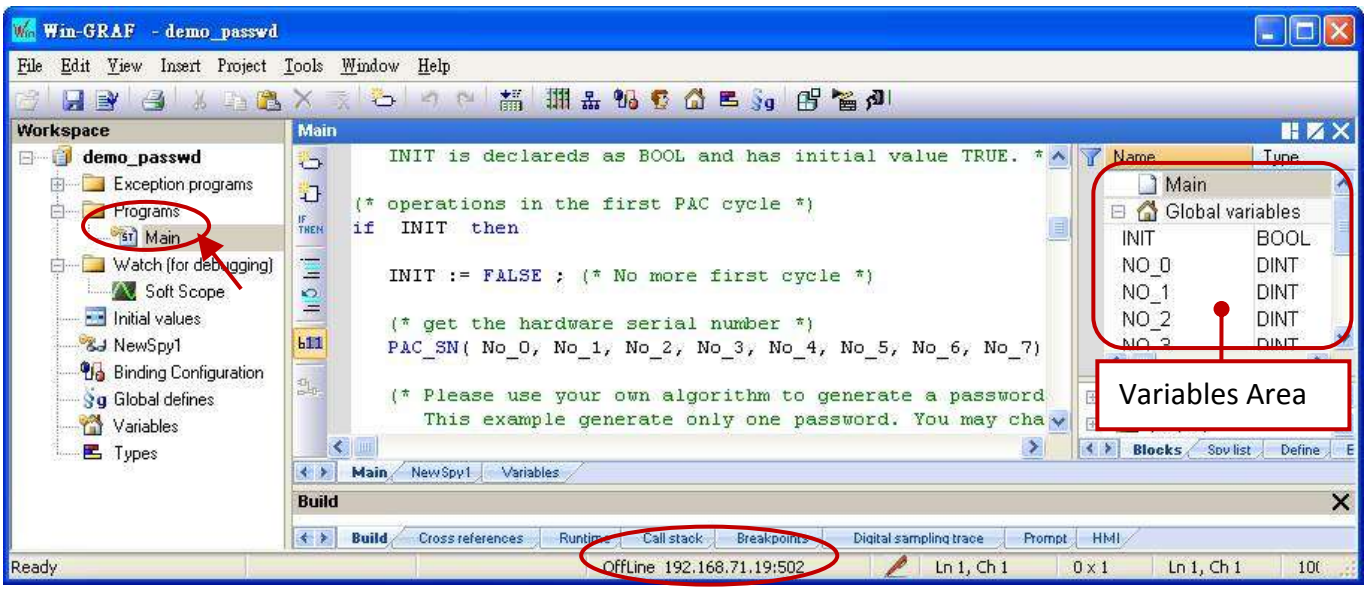
When you finish a Win-GRAF application development and prepare for delivery to the customer, please think about the possibility that your Win-GRAF application on the PAC may will be copied into another same model PAC?! Be careful! Someone else may steal your hard outcome! The following provides a simple and easy way to protect your application.

Note: If you give the Win-GRAF application Source Code to the customer, then sorry, the following method will not protect your program from stolen. Because having the Source Code, anyone can modify the code and apply into another PAC.

Each ICP DAS Win-GRAF PAC has a Serial Number that has 8 Bytes (also known as 64-Bit), and each PAC has the different and unique Serial Number. Therefore, you can use this serial number combine with your own algorithm to generate a password, and pre-store this password on the PAC's file. Then, verify this password in your application. If not passed, the application will not be allowed to execute.

The steps are as follows:
This example uses two Win-GRAF projects, one is "demo_passwd" used to generate a PAC password and save it into the PAC's file; the other is "demo_my_ap" application that has been developed and ready to ship to the customer. Before shipping a PAC to the customer, user needs to download the project "demo_passwd" into the PAC and runs it once to generate a unique password for that PAC. Then, downloads the project "demo_my_ap" into the same PAC, and then can ship the PAC to the customer. After that, if someone copies the Win-GRAF application in this PAC to another same type of PAC and the operation will fail because of the password validation failure.

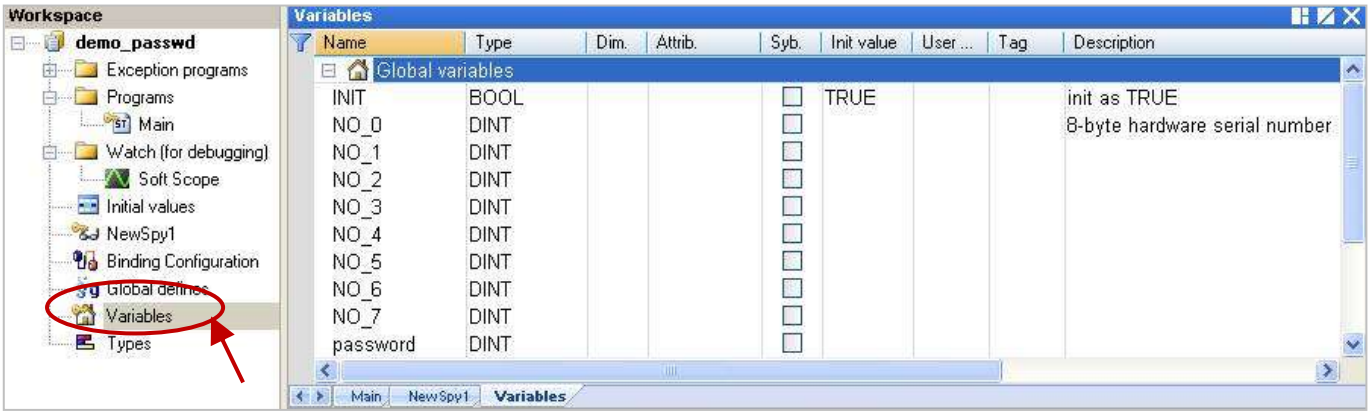
There are two example projects (demo_passwd.zip and demo_my_ap.zip) in the shipment CD (\Napdos\Win-GRAF\demo-project), please refer the Chapter 12 to restore the projects (Execute File > Add Existing Project > From Zip) and set up the IP address of the current PAC.



"demo_passwd" Project:

This program first uses "PAC_SN" Function to read out the Serial Number, and then uses a user-defined algorithm to generate a password. Finally, save the password to an address within the PAC's file (user can decide where you want to store).

Variable Declaration:

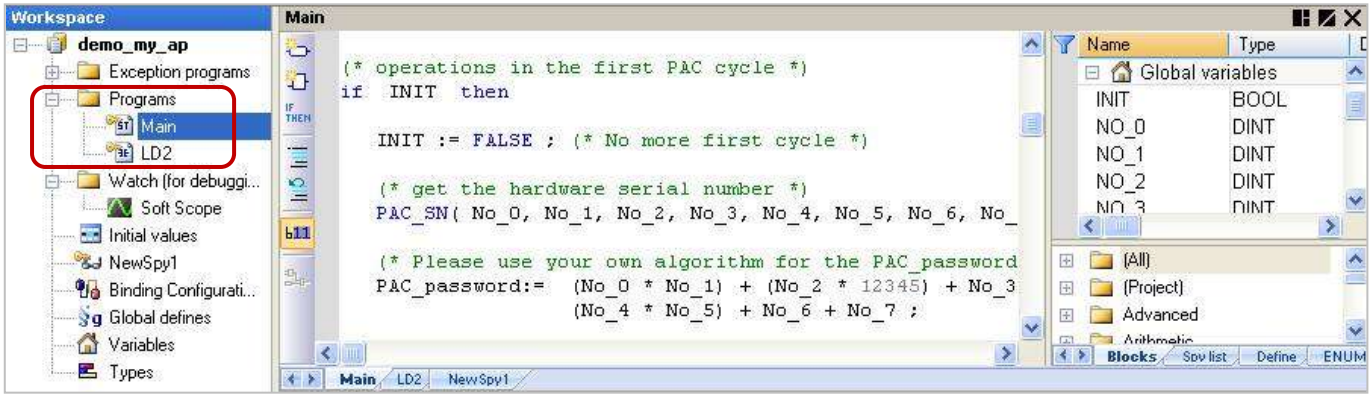


ST Program:

```
(* This "demo_passwd" example will generate a password by the 8-Byte Serial Number of the PAC
   and save it into the EEPROM of the PAC *)

(* Declare "No_0" ~ "No_7" and "password" variables as DINT.
   Declare "INIT" variable as BOOL and has Initial value TRUE. *)

(* Operations in the first PAC Cycle *)
if INIT then
    INIT := FALSE ; (* No more first cycle *)

  (* Get the hardware serial number *)
  PAC_SN( No_0, No_1, No_2, No_3, No_4, No_5, No_6, No_7) ;

  (* Please use your own algorithm to generate a password. This example generate only one
     password. You may change it to generate some passwords. *)
  password := (No_0 * No_1) + (No_2 * 12345) + No_3 + (No_4 * No_5) + No_6 + No_7 ;

  (* save the password in a file "my_product.pwd" in the \System_Disk\Win-GRAF *)
  file_name := '\System_Disk\Win-GRAF\my_product.pwd' ;
  file_id := f_wopen( file_name ) ;
  if file_id = 0 then
    (* failed , do nothing *)
  else
    (* open file ok, save the password into it *)
    fm_write( file_id , Any_to_String(password) ) ;
    f_close( file_id ); (* close file *)
  end_if ;
end_if ;
```

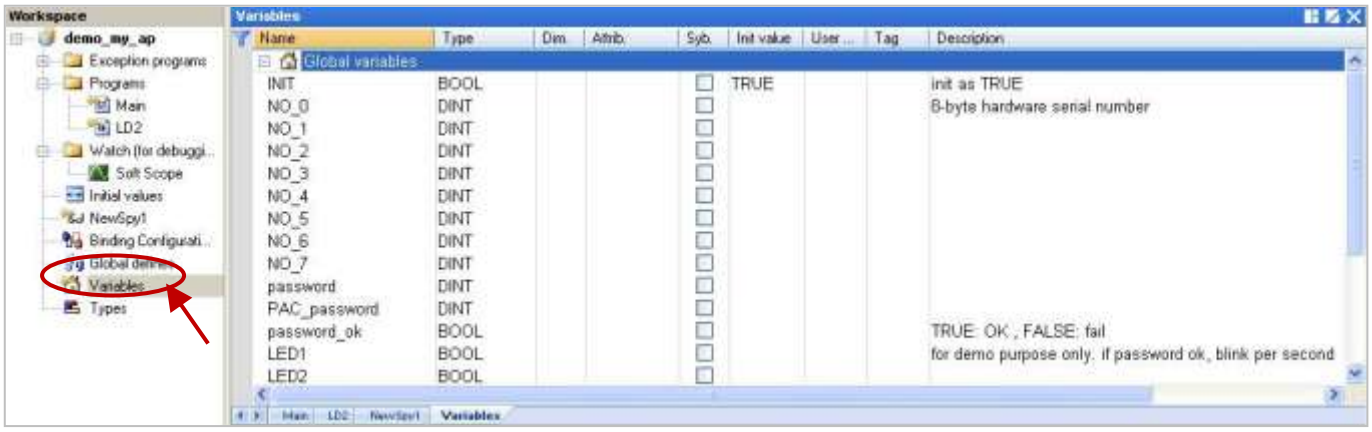
"demo_my_ap" Project:

This project first uses "PAC_SN" Function to read out the Serial Number, then calculates the password, and then compares the password that read from the PAC's file to check if the password correct.



(**Note:** Refer the Section 2.1.2 to arrange the programs in the execution order.)

Variable Declaration:



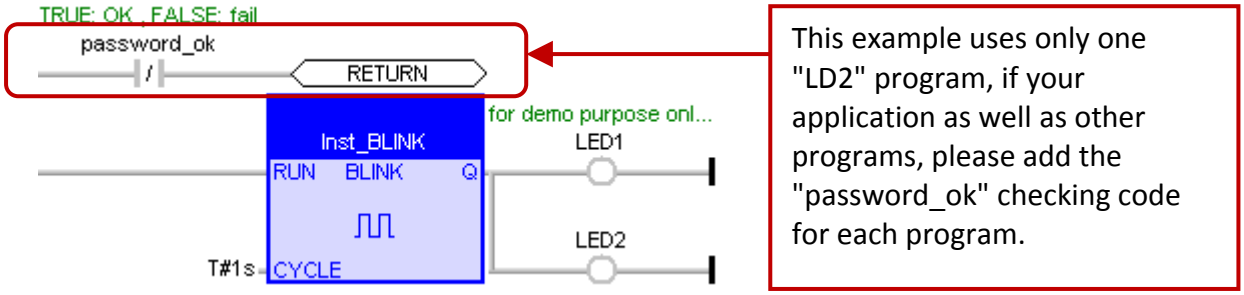
ST Program - Main:

```
(* This "demo_my_ap" example can read the password from the EEPROM of the PAC, and check if
   match with the result that calculated from the user's own algorithm.  *)

(* Declare  "No_0" ~ "No_7", "password" and "PAC_password" variables as  DINT.
   Declare  "INIT" variable as BOOL and has initial value TRUE.
   Declare  "password_ok" variable as BOOL  *)

(*  Operations in the first PAC cycle  *)
if  INIT  then
     INIT := FALSE ;  (*  No more first cycle  *)

  (*  get the hardware serial number  *)
  PAC_SN( No_0, No_1, No_2, No_3, No_4, No_5, No_6, No_7) ;
```

```
  (*  Please use your own algorithm for the "PAC_password" value  *)
 PAC_password:=  (No_0 * No_1) + (No_2 * 12345) + No_3 + (No_4 * No_5) + No_6 + No_7 ;

  (*  Read the password value from a file "my_product.pwd" in \System_Disk\Win-GRAF  *)
  file_name := '\System_disk\Win-GRAF\my_product.pwd' ;
  file_id := f_ropen( file_name ) ;
  if  file_id = 0  then
    (* can not open file, set password to 0 *)
    password := 0 ;
  else
    (* open file ok, read the password *)
    if  f_eof( file_id )  then
      (* reach the end of file *)
    else
      (* hasn't reached the end of file , read a string form it *)
      Tmp_string := fm_read( file_id ) ;
      (* Convert a string to a DINT value *)
      password := Any_to_DINT(Tmp_string) ;
    end_if ;
    f_close(file_id) ;  (* close file *)
  end_if ;

  (*  check if the password is correct?  *)
  password_ok := FALSE ;  (*  set it as "FALSE" in the beginning *)
  if  password = PAC_password  then
    password_ok := TRUE ;   (*  the password is correct  *)
  end_if ;

end_if ;
```

LD Program – LD2

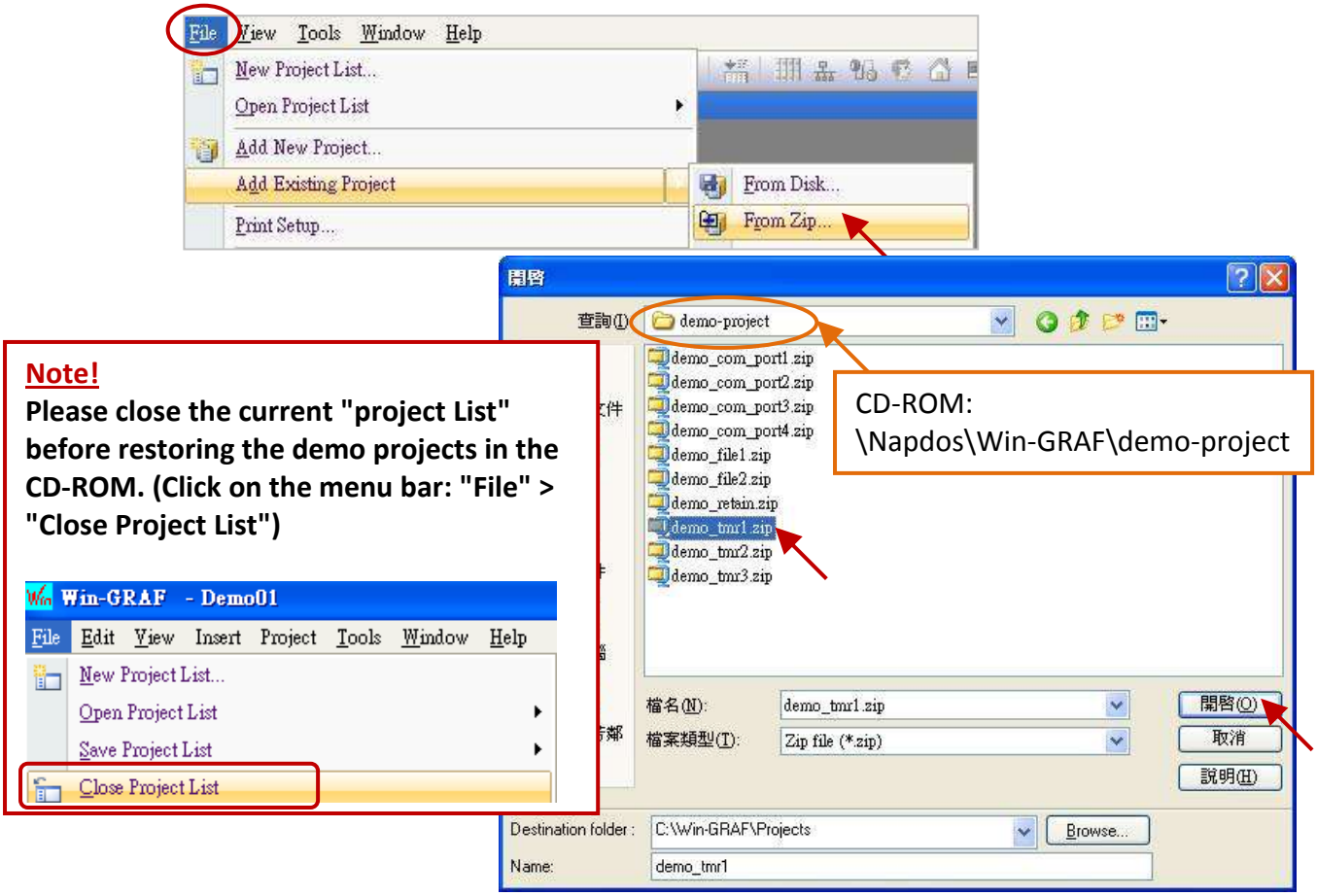
If the "password_ok" is “FALSE“, it means the password is incorrect, then will exit the program. Only when the password is correct can the program execute continuously, and then your application can be protected from the unauthorized access.



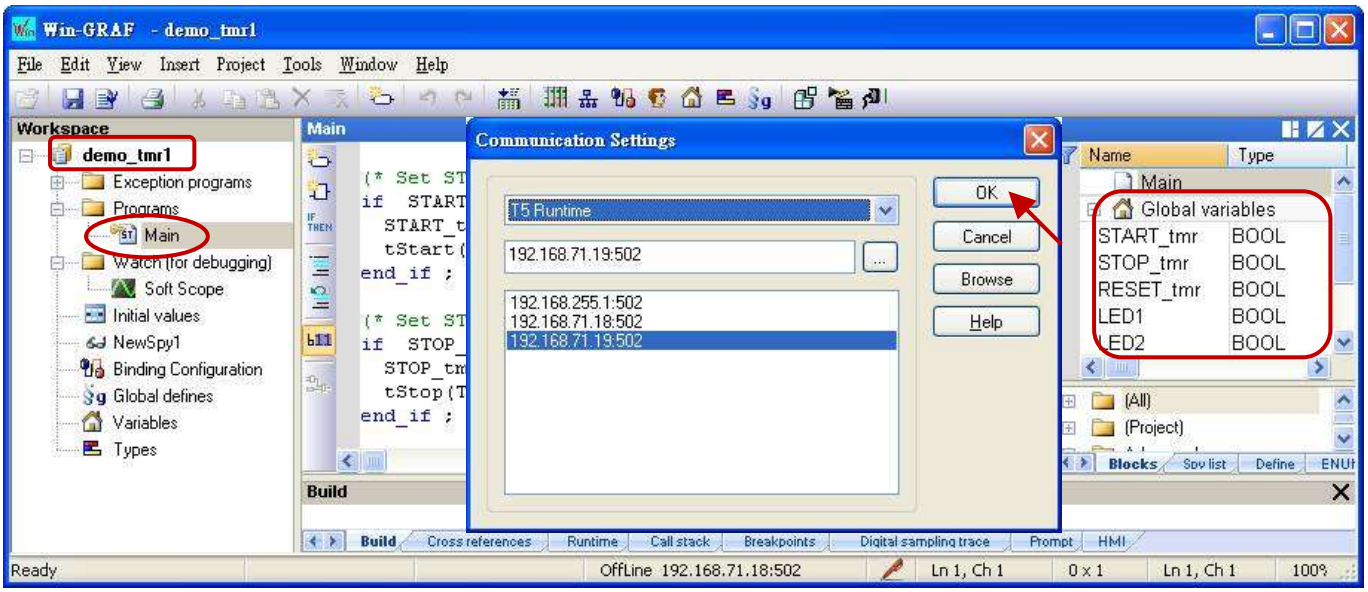
Chapter 12 Description of Win-GRAF Demo Projects

There are some demo projects in the Win-GRAF shipping CD (CD-ROM: **\Napdos\Win-GRAF\demo-project**) which some of them will be introduced in the following sections.
Before using the demo projects, follow the steps below:

1. Click on the menu bar "File" > "Add Existing Project" > "From Zip" to open a project (e.g., "demo_tmr1.zip").



2. Double click "Main" to open the ST program, and can view/add variables in the variable area.
3. Mouse right-click on the project name ("demo_tmr1"), and select "Communication Parameters" to set up the IP Address of your PAC. (Refer Section 2.3.5)



12.1 The List of Demo Programs

The Win-GRAF product package includes a CD-ROM named Win-GRAF-PAC-CD, the user can find out more information on Win-GRAF demo programs in the path - \Napdos\Win-GRAF\demo-project. Refer the table below to restore these programs to the Win-GRAF Workbench.

The following table shows the list of the demo programs:

File Name	Description
demo_tmr1	Use the "tStart" and the "tStop" functions to operate the Timer (refer section 12.2.1).
demo_tmr2	To do periodic operations for the Timer (refer section 12.2.2).
demo_tmr3	To do periodic operations more accurately for the Timer (refer section 12.2.2).
demo_com_port1	Send a string by the COM Port (refer section 12.3.1).
demo_com_port2	Request/Anser the device by the COM Port (refer section 12.3.2).
demo_com_port3	Wait for data coming from the remote device to the COM Port (refer section 12.3.3).
demo_com_port4	Report data periodically to the remote device by the COM Port (refer section 12.3.4).
demo_file1	Write data to a file on the PAC (refer section 12.4.1).
demo_file2	Read data from a file on the PAC (refer section 12.4.2).
demo_retain	Use retain variables to save lately values before the PAC power failure (refer section 6.1).
demo_wp5_retain	Use file to retain variable values (refer section 6.2).
demo_extra_port	Enable a serial port for connecting the Win-GRAF Workbench (refer appendix E).
demo_my_ap	Protect your Win-GRAF program and avoid being embezzled (refer section 11.12).
demo_passwd	
demo_XV310	Use an XV-Board in the WP-5xx8-CE7 (refer section 5.1.6 ~ 5.1.12).
demo_XV308_1	
demo_XV308_2	
demo_XV308_3	
demo_XV116	
demo_XV111	
demo_XV110	
demo_XV107	
demo_vb01	Use VB.net 2008 programs to read/write Win-GRAF variables (refer Chapter 13).
demo_vb02	
demo_vb03	
demo_vb04	
demo_PID_simple	The PID and regulator applications.
demo_user_C	Develop your own Functions and Function Blocks (refer Chapter 18).

File Name	Description
demo_rdn_1	Redundant System (refer Chapter 16).
demo_rdn_2	
demo_rdn_3	
demo_ET7060	Connect the remote ET-7060 Ethernet I/O module (refer section 5.2.2).
demo_ET7018z	Connect the remote ET-7018Z Ethernet I/O module (refer section 5.2.3).
DEMO_DL_100T485	Connect the remote DL-100T485 module to measure temperature and humidity (refer section 8.2.7).
demo_schedule	Schedule control (refer Chapter 17).
demo_d_7065	Connect the remote I-7065 I/O module (refer section 8.2.1).
demo_d_7018z	Connect the remote I-7018Z I/O module (refer section 8.2.2).
DEMO_D_7083	Connect the remote I-7083 I/O module (refer section 8.2.3).
demo_8088w	PWM Output (refer section 4.11).
DEMO_D_87084_FR	Connect the remote I-87084W module to measure frequency (refer section 8.2.4).
DEMO_D_87084_C4	Connect the remote I-87084W module to measure Counter (refer section 8.2.5, 8.2.6 and 4.9).
DEMO_D_87084_C8	
demo_SMS	Use the 2G/3G GSM Modem to send/receive the text message (refer Chapter 21).
demo_send_file	Send a file via Ethernet or 3G wireless network from a PAC to a remote PC (refer Chapter 20).
demo_3G	3G wireless communication.

12.2 Timer Operations

12.2.1 Start, Stop and Reset the Timer

Refer P12-1 to open the project ("demo_tmr1.zip"), and can view/add variables in the variable area.

ST Program:

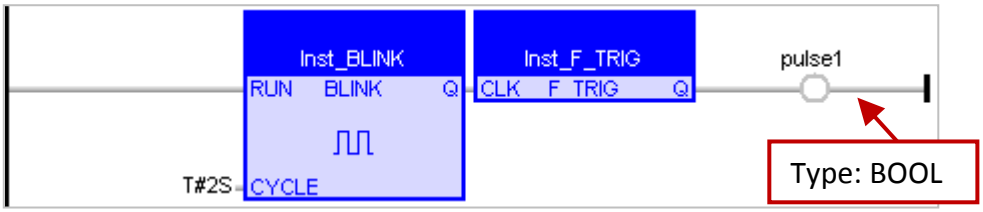
```
(* Declare "START_tmr", "STOP_tmr", "RESET_tmr", "LED1", "LED2" as BOOL
   Declare "TMR1" as TIME *)
(* Set START_tmr as TRUE to start ticking Timer TMR1 *)
 IF   START_tmr   THEN
   START_tmr := FALSE ;
   TSTART (TMR1) ;
END_IF;

(* Set STOP_tmr as TRUE to stop ticking Timer TMR1 *)
 IF   STOP_tmr   THEN
    STOP_tmr := FALSE ;
    TSTOP (TMR1) ;
 END_IF;

(* Set RESET_tmr as TRUE to reset TMR1 to a value T#0s *)
 IF   RESET_tmr   THEN
    RESET_tmr := FALSE ;
    TMR1 := T#0s ;
 END_IF;

(* Let LED1, LED2 ON during TMR1 = 3 ~ 10 second *)
 LED1 := FALSE ;
 LED2 := FALSE ;
 IF   (TMR1 >= T#3s) and (TMR1 <= T#10s) THEN
      LED1 := TRUE ;
      LED2 := TRUE ;
 END_IF;

(* Reset TMR1 as 0 when reachs 15 second *)
 IF   TMR1 >= T#15s THEN
      TMR1 := T#0s ;
 END_IF;
```

12.2.2 Periodic Operations

Refer P12-1 to open the project ("demo_tmr2.zip"), and can view/add variables in the variable area.

Function "BLINK" plus Function Block "F_TRIG" can produce a Pluse TRUE at regular intervals, so it can be applied in the periodic operations.

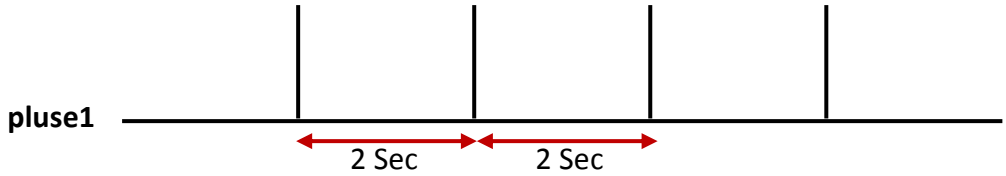
LD Program:



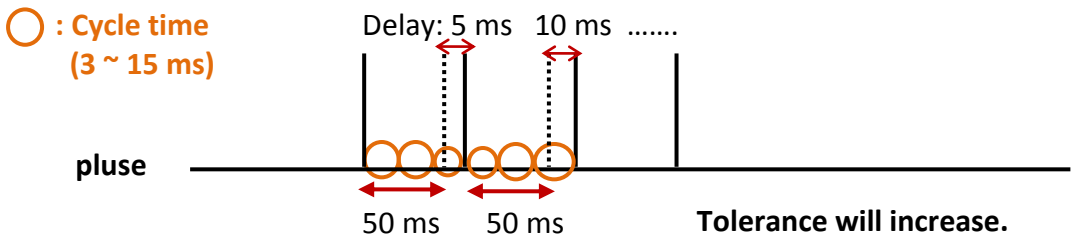
ST Program:

```
IF   pluse1   THEN
  (*  do periodic operations here *)
  • • •
END_IF;
```

The "BLINK" and "F_TRIG" above will produce a Pluse TRUE every two seconds, but this method has a drawback:



If the interval time of the period is shorter (e.g., 100 ms per period or less; or the PAC Cycle Time is larger, such as 20 ~ 50 ms, generally 3 ~ 15 ms), then the period operations will be inaccurate. For example, to do a periodic operation every 50 ms, as compared to 250 ms or 2 seconds, the interval time of 50 ms is very close to the PAC Cycle Time, if using Function Blocks **"BLink" plus "F_TRIG"**, it is easy to accumulate the output delay time, therefore the final operation time will become inaccurate.



To improve the situation above, the following coding method will be more accurate:
(**Tolerance will not increase.**)

Refer P12-1 to open the project ("demo_tmr3.zip"), and can view/add variables in the variable area.

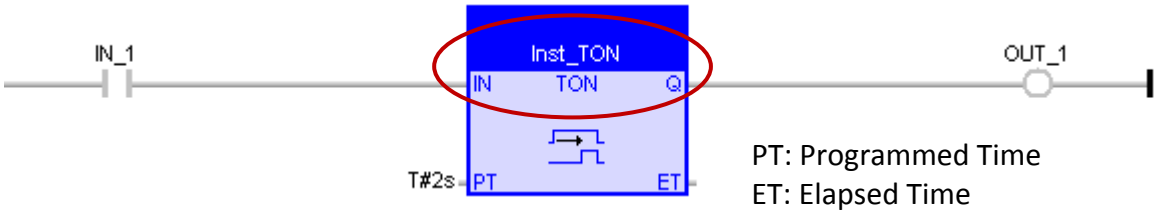
ST Program:

```
(* Declare "INIT"  as BOOL and has initial value  TRUE
   Declare "TMR1",  "TMR1_next" as TIME  *)

 IF   INIT   THEN
    INIT  :=  FALSE ;
    TMR1  :=  T#0s ;
   TMR1_next  :=  TMR1  +  T#50 ms ;
   TSTART (TMR1);
END_IF;

IF   TMR1  >=  TMR1_next   THEN
    IF   TMR1  >  T#10h  THEN
        TMR1  :=  T#0s ;
        TMR1_next  :=  T#0s ;
   END_IF;
   TMR1_next  :=  TMR1_next  +  T#50 ms ;
   (*  Do periodic operations here *)
   • • •
END_IF;
```

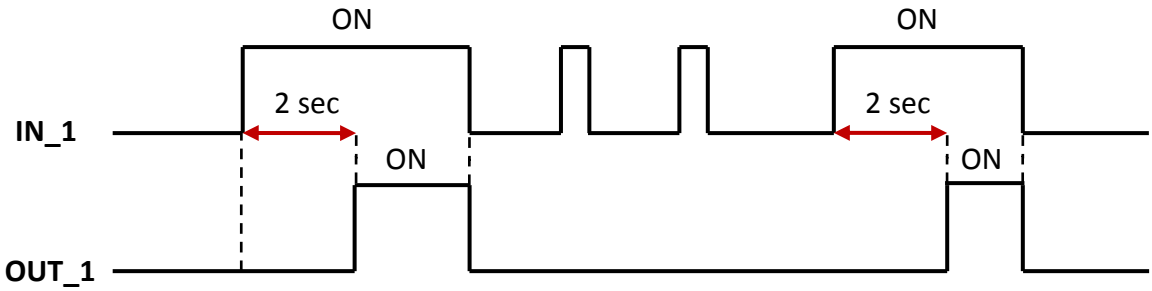
When the timer reach T#23h59m59s999ms, the value will overflow.
Therefore, please reset it automatically to "0" after 10 or 18 hours.

12.2.3 Detect the Steady ON or Steady OFF Signal

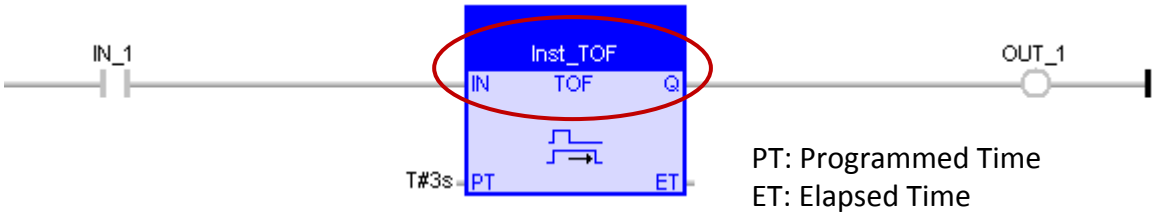
"TON" Function Block can detect the steady "**ON**" signal. (Keeps "**ON**" for a minimum period of time.)



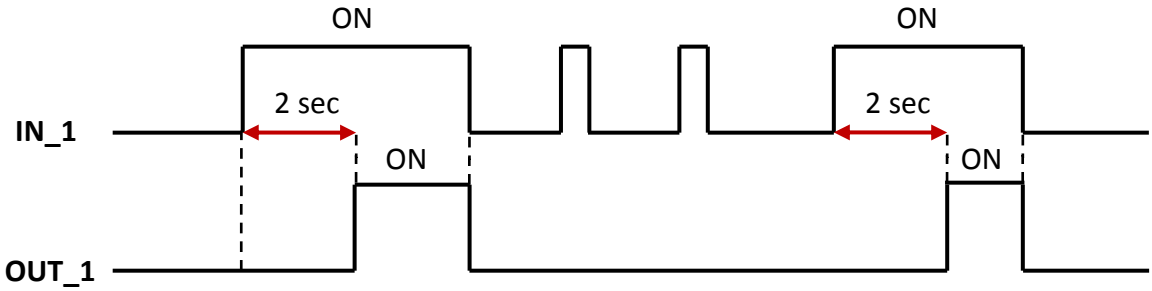
As the picture above, this function can detect the steady "**ON**" signal that keep at least 2 seconds.

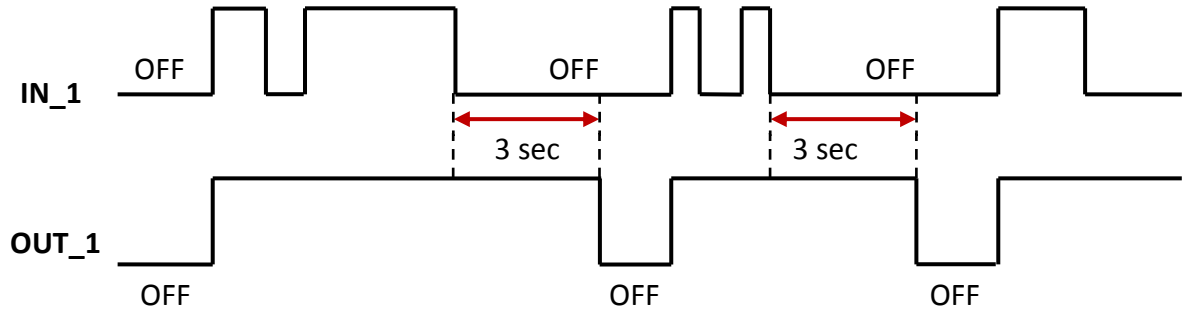


"TOF" Function Block can detect the steady **"OFF"** signal. (Keeps **"OFF"** for a minimum period of time.) that can keep **"OFF"** for a period of time.



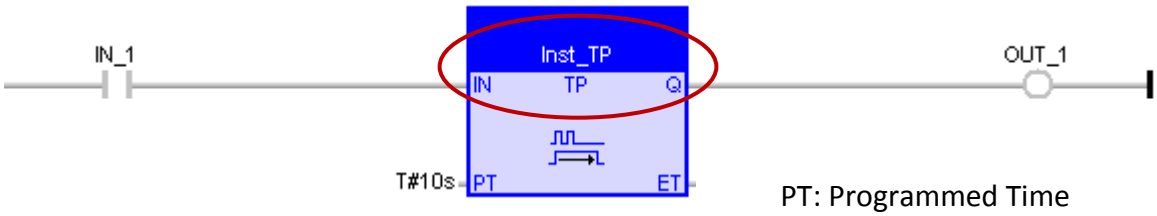
As the picture above, this function can detect the steady **"OFF"** signal that keep at least 3 seconds.





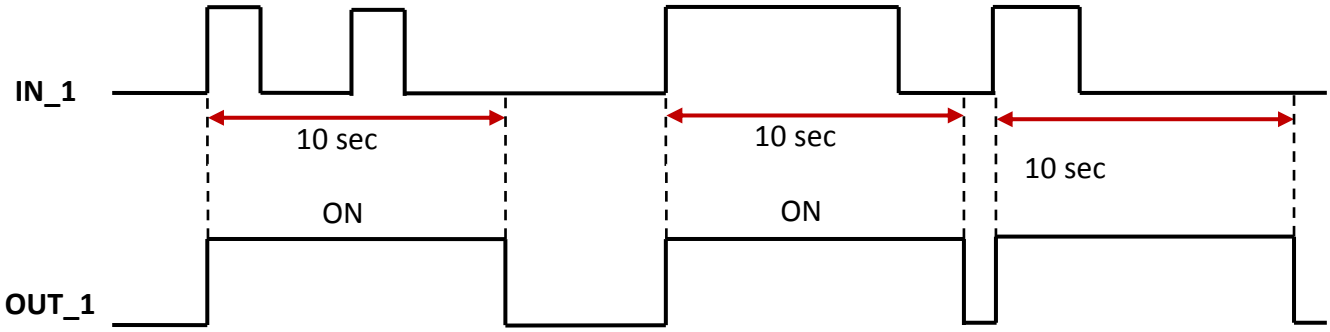
12.2.4 Keep Outputting ON for Some Time after Triggering

"TP" Function Block can keep outputting "ON" for some time after triggering (e.g., from OFF to ON).



PT: Programmed Time
ET: Elapsed Time

As the picture above, after triggering, it can keep outputting **"ON"** signal for 10 seconds.

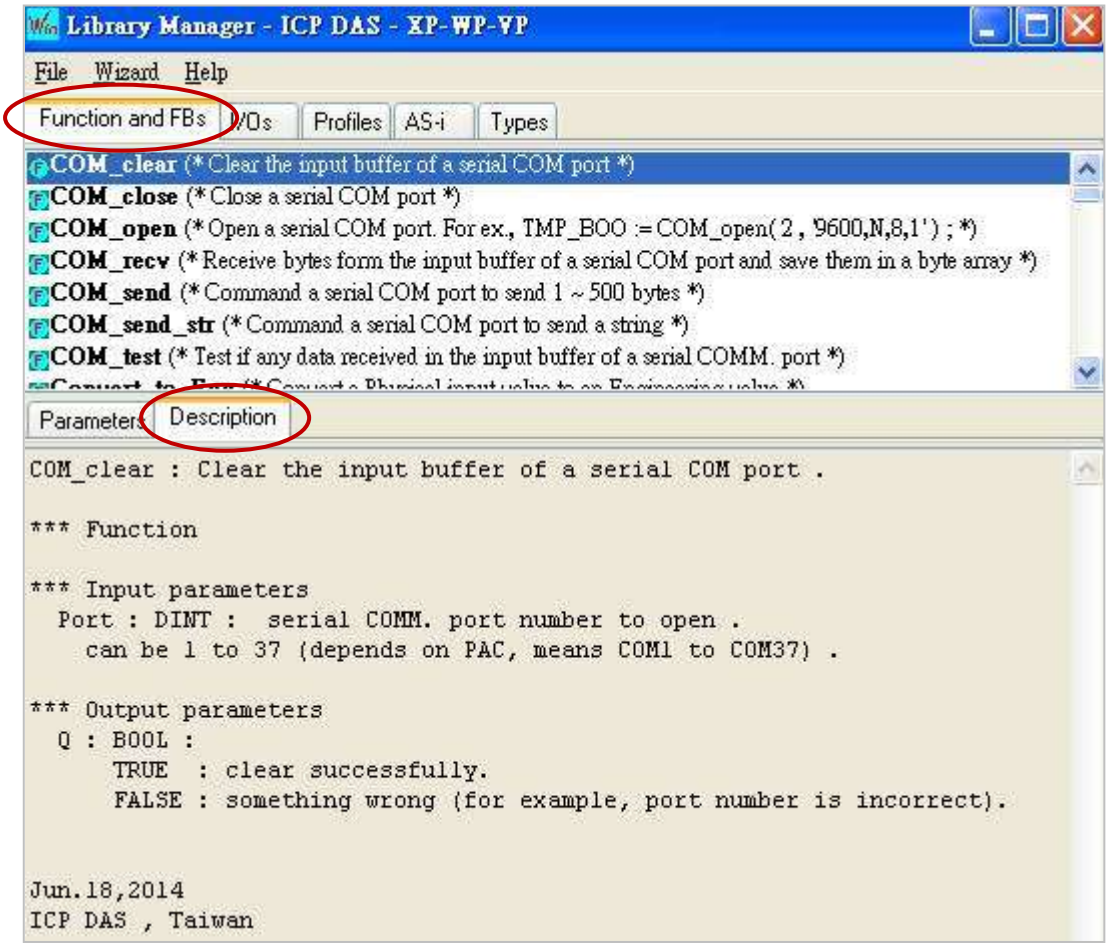


12.3 Operations of Serial Port Communication

Users can directly operate the serial port (e.g., RS-232, RS-485 or RS-422 Port) to achieve some specific communication protocol. The following Functions can be used to directly operate the serial port.

Functions	Description
COM_open	Open a serial COM port.
COM_close	Close a serial COM port.
COM_clear	Clear the input buffer of a serial COM port.
COM_test	Test if any data received in the input buffer of a serial COMM port.
COM_send	Command a serial COM port to send 1~500 bytes.
COM_send_str	Command a serial COM port to send a String.
COM_recv	Receive bytes from the input buffer of a serial COM port and save them in a byte array.
COM_status	Get the current status of a serial COM port.

Please refer Section 1.2.3 to open the Library Manager and find the detail Function description.



12.3.1 Send a String by the COM Port

Refer P12-1 to open the project ("demo_com_port1.zip"), and view/add variables in the variable area.

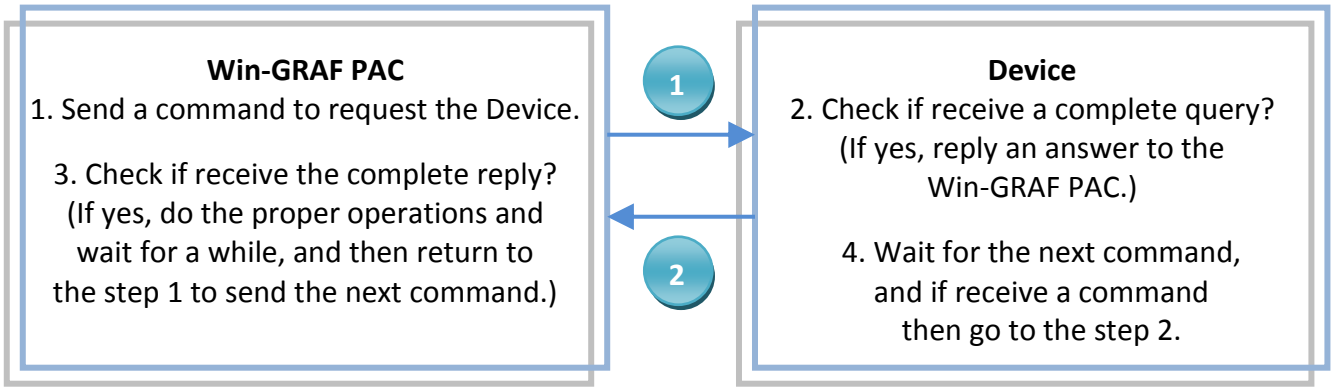
ST Program: This program can send a String every 2 seconds by the PAC COM1 (parameters: `9600,N,8,1') (e.g., < CNT1 = 1 > or < CNT1 = 25 >).

```
Declare "INIT" as BOOL and has initial value TRUE;
        "Port_OK" as BOOL ;
        "CNT1" as DINT ;
        "TMR1", "TMR1_next" as TIME
        "Port_number" as DINT and has initial value "1"
```

```
  (* Operations in the first PAC cycle *)
  if INIT then
    INIT := FALSE ; (* No more first cycle *)
    CNT1 := 0 ;
    TMR1 := T#0s ;
    TMR1_next := TMR1 + T#2s ;
    (* start ticking TMR1 *)
    tStart(TMR1) ;
  end_if ;

    (* if the status of COM port becomes FALSE(not open), open it *)
  if COM_Status(Port_number) = FALSE then
    (* open a serial COM port *)
    Port_OK := COM_open(Port_number, '9600,N,8,1' ) ;
  end_if ;

    (* when time reached , ... *)
  if TMR1 >= TMR1_next then

    (* to prevent TMR1 overflow (means reach T#23h59m59s999ms) *)
    if TMR1 > T#10h then
      TMR1 := T#0s ;
      TMR1_next := T#0s ;
    end_if ;

    (* Set new TMR1_next *)
    TMR1_next := TMR1_next + T#2s ;

    (* Send a string from COM port *)
    COM_send_str( Port_number, '<CNT1=' + Any_to_STRING(CNT1) + '>' ) ;

    (* reset CNT1 when reach 100 *)
    CNT1 := CNT1 + 1 ;
    if CNT1 >= 100 then
      CNT1 := 0 ;
    end_if ;
end_if;
```

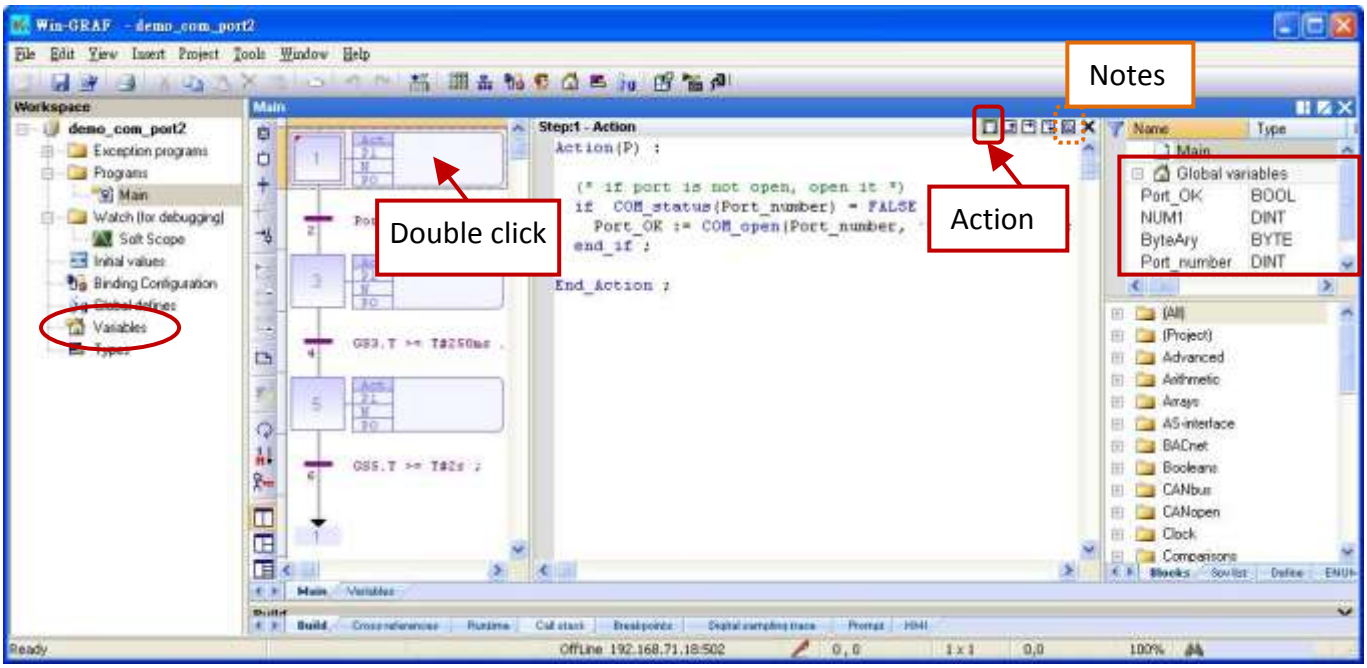
12.3.2 Request/Answer the Device by the COM Port

If an application needs to use RS-232/485/422 Port to get the data from other devices, the steps are as the following request and answer:

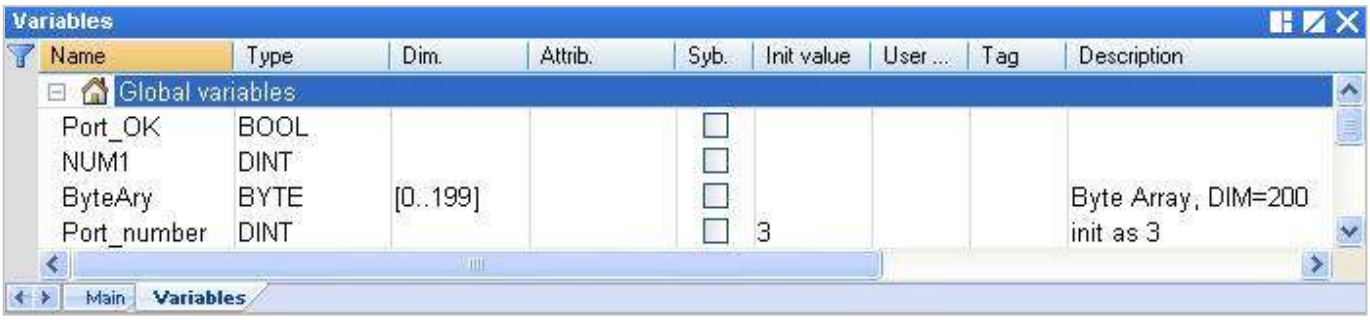


Refer P12-1 to open the project ("demo_com_port2.zip"), and view/add variables in the variable area.

Note: Double click "Action" will first open a "Notes" window, then click "Action" icon to see the code.



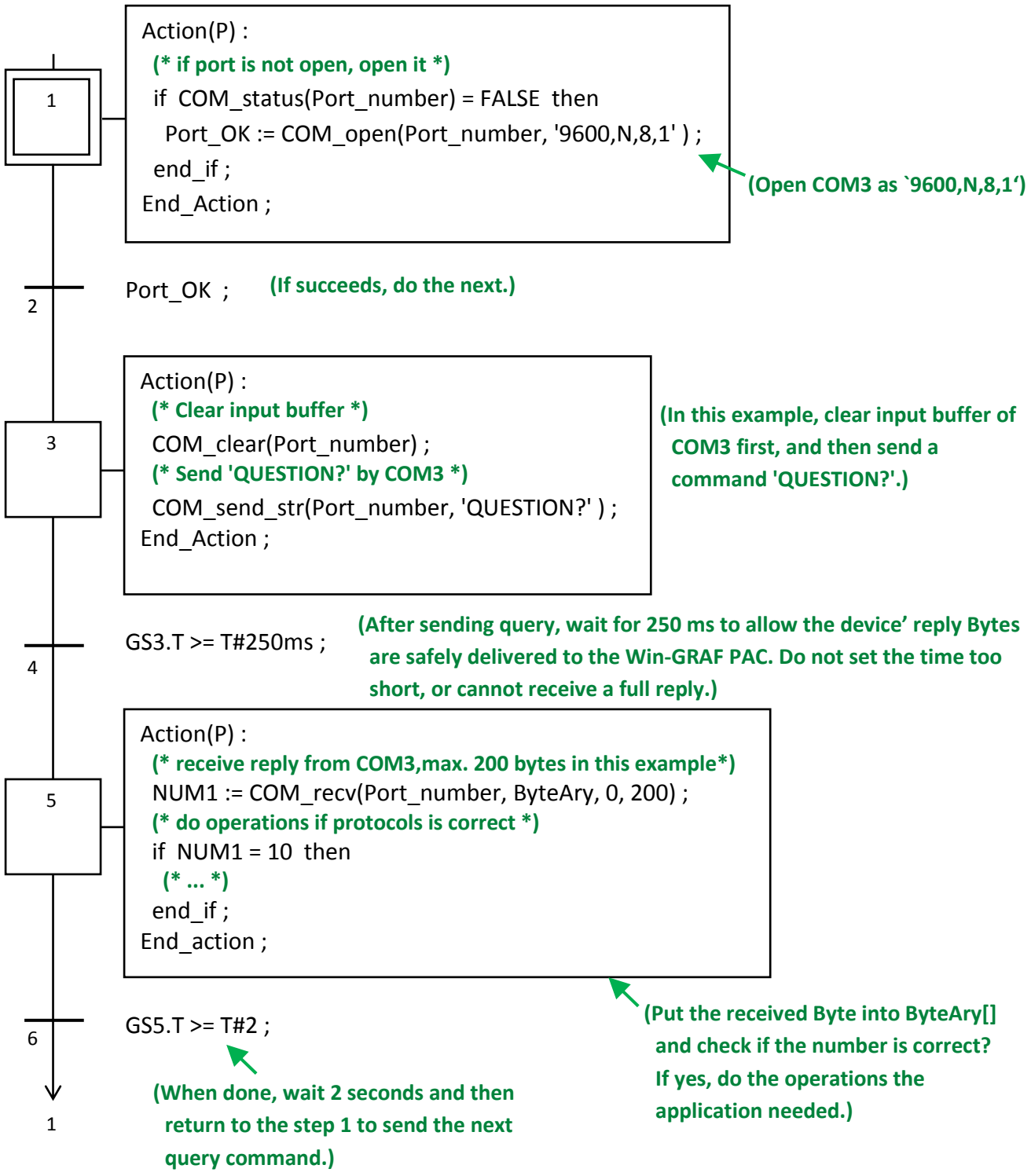
You can click the "Variables" tag to open the Variables window.



In this example, the Win-GRAF PAC sends a string 'QUESTION?' by COM3 to the device, and then wait for the reply and does operations. After the operations, waits 2 seconds, and then sends the same command 'QUESTION?', and repeated.

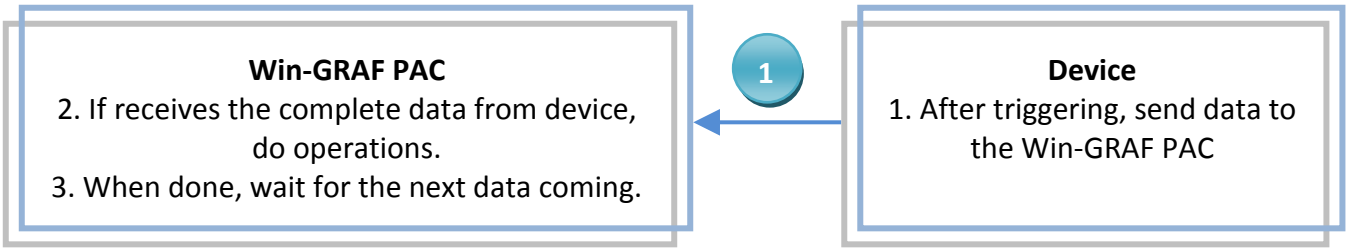
SFC Program:

(Declare "Port_OK" as BOOL ; "NUM1" as DINT ; "ByteAry" as BYTE and Dim. = "200" ;
"Port_number" as DINT and has initial value "3")



12.3.3 Wait for Data Coming from the Remote Device to the COM Port

This way is common in the general store or supermarket, such as using the barcode readers. After reading the barcode of the product, it will send the barcode data to the Win-GRAF PAC's COM Port (RS-232/485/422), and need not to reply any messages.



Refer P12-1 to open the project ("demo_com_port3.zip"), and view/add variables in the variable area.

ST Program:

```
(* operations in first PAC cycle *)
if  INIT  then
  INIT := FALSE ;
  T1 := T#0s ;
  STEP1 := 0 ;
end_if ;
```

Declare "INIT" as BOOL and has initial value TRUE;
"Port_OK" as BOOL ;
"STEP1", "NUM1" as DINT ;
"T1" as TIME ;
"ByteAry" as BYTE and Dim. = "200" ;
"Port_number" as DINT and has initial value "3".

```
(* if port is not open, open it *)
if  COM_status(Port_number) = FALSE  then
  Port_OK := COM_open( Port_number , '9600,N,8,1' ) ;
end_if ;

(* If open port fail, exit this ST program *)
if  Port_OK = FALSE  then
  return ;
end_if ;

CASE  STEP1  OF

  (* if there is at least 1 byte coming *)
  0:
     if  COM_test(Port_number)  then
       STEP1 := 1 ;
       T1 := T#0s ;
       Tstart(T1) ;
     end_if ;
```

STEP1 = 0, means waiting, and will test if COM3 has data?
If returns TRUE, means COM3 has data.
Then set STEP1 to "1", T1 to "0" and start timing.

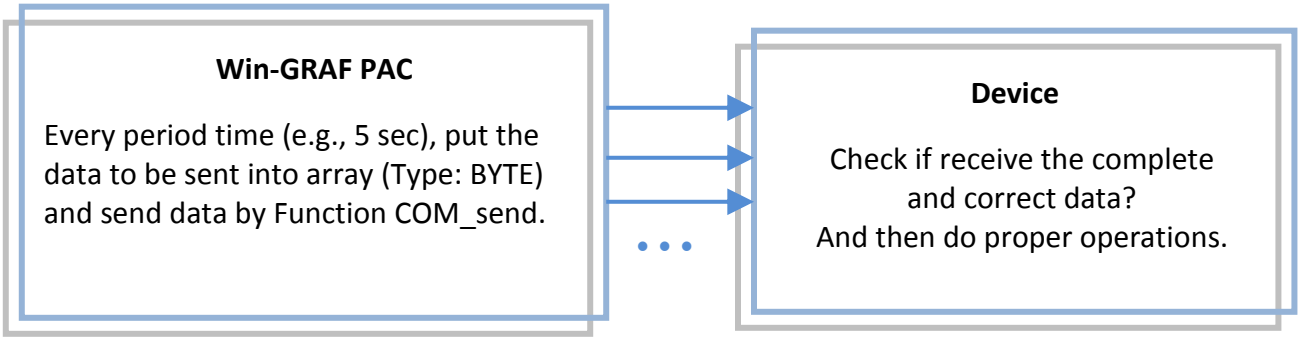
```
 (* wait 250 ms, then receive all bytes form COM port *)
1:
  if  T1 >= T#250ms  then
    Tstop(T1) ;
    T1 := T#0s ;
    STEP1 := 0 ;

    (* receive max. 200 bytes *)
    NUM1 := COM_recv(Port_number , ByteAry , 0 , 200 ) ;

    (* do proper operations if protocol is correct ,
        here assume correct protocols has 25 bytes in this example*)
    if  NUM1 = 25  then
     (* ... *)
    end_if ;

  end_if ;

END_CASE ;
```

STEP = 1: means the data is sending in, and will wait 250 ms to receive all data and put them into an array. The waiting time concerns the device specifications and the Baud Rate. If set the time too short, may receive data incompleted. Remember to set STEP1 to "0" to wait for the data coming next time.

When receive data, check if data is correct?
If yes, do the operations the application needed.

12.3.4 Report Data Periodically to the Remote Device by the COM Port

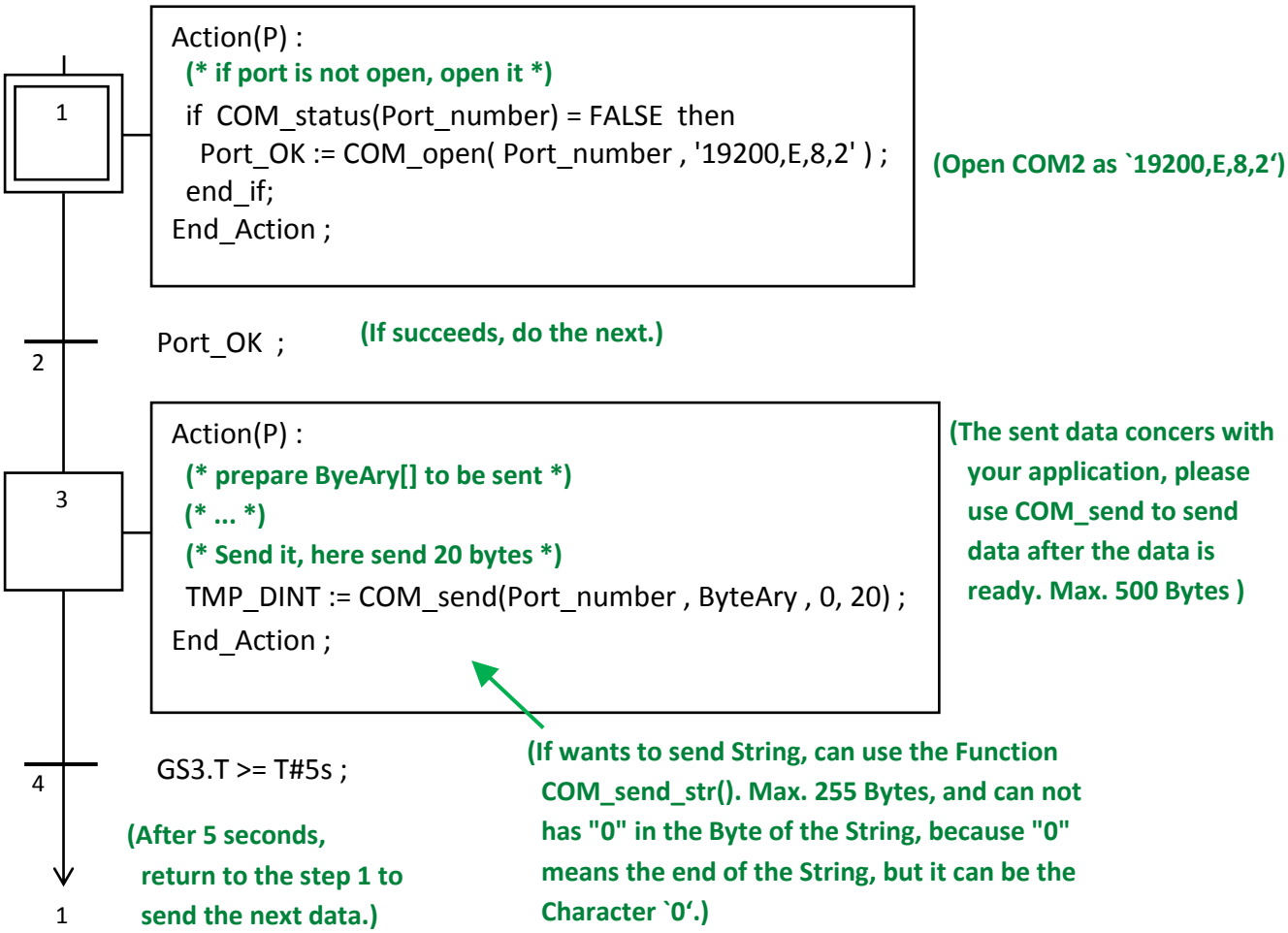
If wants to periodically report data to other devices by RS-232/485/422 Port, do as follows.



Refer P12-1 to open the project ("demo_com_port4.zip"), and view/add variables in the variable area.

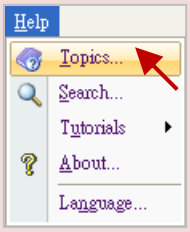
SFC Program: (Refer Section 12.3.2 to open the "Action" window.)

(Declare "Port_OK" as BOOL ; "TMP_DINT" as DINT ; "ByteAry" as BYTE and Dim. = 100 ; "Port_number" as DINT and has an initial value "2".)



12.4 Read/Write Data from/to a File on the PAC

The Win-GRAF Workbench provides the following Functions to enable sequential read/write operations in disk files of the Win-GRAF PAC.

Functions	Descriptions
Please click the menu bar "Help" > "Topics" and type the searching key word "File" to see more detail information in the topic of the "File Management functions". 	
F_ROPEN	Open/Create a file for reading.
F_WOPEN	Open/Create a file for writing.
	If the file doesn't exist, it will be created automatically. If the file exists, all its content will be removed.
F_AOPEN	Create or open a file in append mode.
F_CLOSE	Close an open file.
F_EOF	Test if the end of file is reached in a file open for read.
FA_READ	Read a DINT integer from a binary file.
FA_WRITE	Write a DINT integer to a binary file.
FM_READ	Read a STRING value from a text file
FM_WRITE	Write a STRING value to a text file.
FB_READ	Read binary data from a file.
FB_WRITE	Write binary data to a file.
F_EXIST	Test if a file exists.
F_GETSIZE	Get the size of a file.
F_COPY	Copy a file.
F_DELETE	Remove a file.
F_RENAME	Rename a file.
Refer Section 1.2.3 to find the detailed descriptions for the following Functions.	
F_dir	Create a directory.
F_cp_dir	Copy all files in a directory to another directory. (Include subdirectories and files). **(Note1)**
F_del_dir	Delete a directory and all files inside it. (Include subdirectories and files). **(Note1)**

Note: **The Win-GRAF PAC of ICP DAS does not support Functions "F_SAVERETAIN" and "F_LOADERETAIN".**

Note1: Since the following PAC drivers support the **"F_cp_dir"** and the **"F_del_dir"** functions for operating in sub-directories.

WP-8xx8 : v1.04 , VP-x2x8-CE7 : v1.01 , XP-8xx8-CE6 : v1.02 , WP-5xx8-CE7: Released day.

12.4.1 Write Data to a File on the PAC

Refer P12-1 to open the project ("demo_file1.zip"), and can view/add variables in the variable area.

ST Program: This program can be used to write 10 "REAL" values to a file on the PAC.

```
(*  This "demo_file1" project will save 10 REAL value to a file
                          in the  \System_Disk\Real_data1.txt .

      File Format :
        Each row contains one REAL value and ends with <CR><LF> characters.  Like:

        1.08
        2.786
        38.45
        41.5
        59.875
        60.76
        71.23
        80.5
        99.8
        100.7        *)

(*  Variables declaration:
     Write_File       : BOOL
     Tmp_string       : String, len=255
     File_ID          : DINT
     REAL_val[0..9] : REAL
     ii                : DINT
     File_Status      : String, len=128    *)

(* Set Write_File as TRUE to write data to the file *)
if  Write_File  then

  Write_File := FALSE ;
  File_ID := F_Wopen( '\System_Disk\Real_data1.txt' );

 if  File_ID = 0  then
    (* Can not open file in write mode *)
    File_Status := 'Can not open file in write mode !' ;
  else
      (* open file in write mode ok, save REAL[0] ~ [9] to file ,
            each row contains 1 REAL value and end with <CR><LF> *)
    File_Status := 'Open file ok.' ;
    for ii := 0 to 9  by 1  do
```

Because the size of \System_Disk\ is small, recommend you may change the directory to below (Depends on your application):

WinPAC, ViewPAC Series:
\Micro_SD or

XPAC Series:
\System_Disk2

```
      Tmp_string := Any_to_string( REAL_val[ii] ) ;
      FM_write( File_ID , Tmp_string ) ;
    end_for ;

    (* close the file *)
    F_close( File_ID ) ;

  end_if ;
end_if ;
```

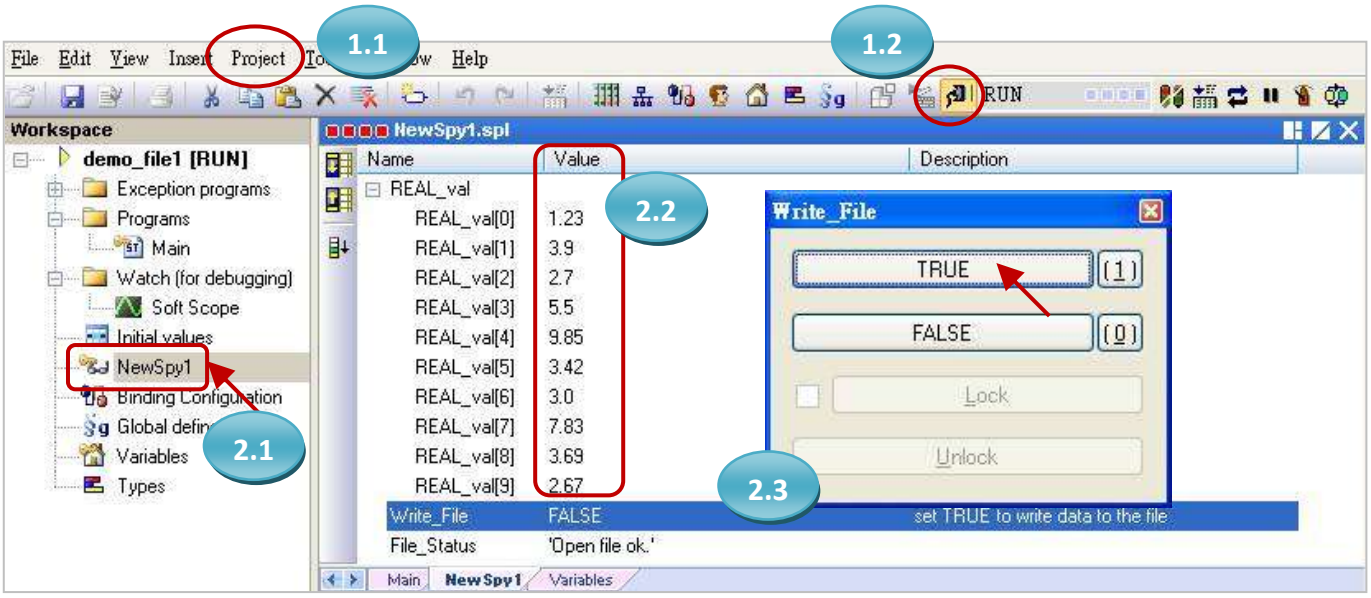
If want to save data to Integer, use the code: Tmp_string := Any_to_string(**DINT_val[ii]**) ; and declare Variable "DINT_val" as DINT and Dim. at least "10" for this example.

Test Program:

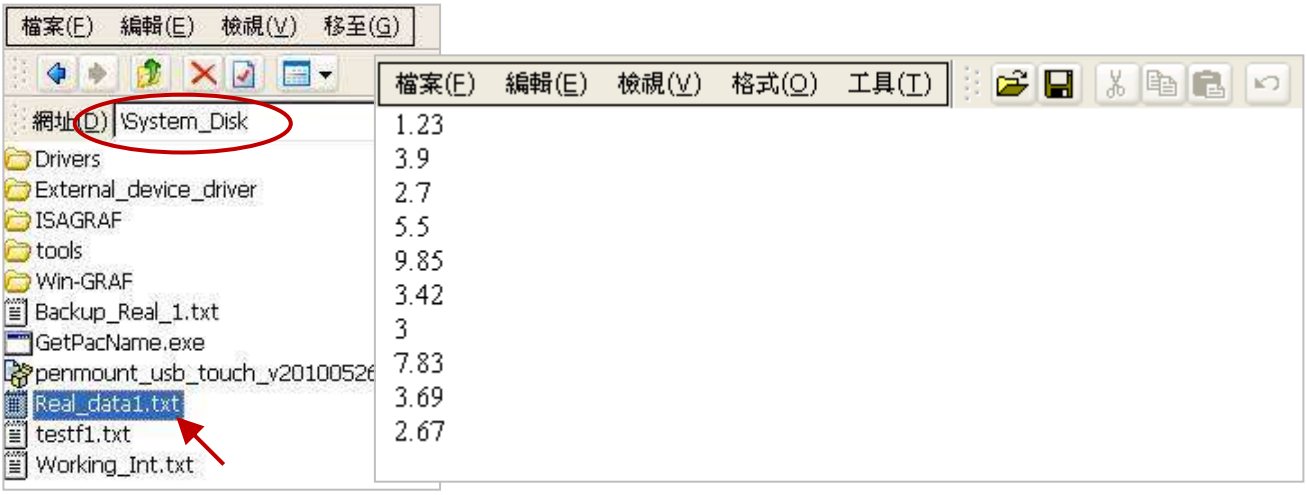
In this example, when the "Write_File" is set to "TRUE", the values will be written into the file \System_Disk\Real_data1.txt on the PAC.

1. Please set up IP configurations (Refer P12-1), compile and download the program to the PAC. (Click on "Project" > "Build All Projects" / "On Line", if not familiar with the operation, refer to Section 2.3.4 , Section 2.3.5)

2. Click "NewSpy1" to open a Spy List and fill in the values to be written, and then set the "Write_File" become "TRUE" to Write data. (If OK, "File_Status" will show "Open file ok".)



3. On the PAC, open the file "Real_data1.txt", can see the values filled in the step 2.



12.4.2 Read Data from a File on the PAC

Refer P12-1 to open the project ("demo_file2.zip"), and can view/add variables in the variable area.

ST Program: This program can be used to read 10 "REAL" values from a file on the PAC.

```
(*   this "demo_file2" project will read 10 REAL value from a file
                                in the  \System_Disk\Real_data2.txt .
      File format :
        Each row contains one REAL value and ends with <CR><LF> characters. Like:
        1.08
        2.786
        38.45
        41.5
        59.875
        60.76
        71.23
        80.5
        99.8
        100.7
*)

 (*
  Variables Declaration:
  Write_File       : BOOL
  Tmp_string       : String, len=255
  File_ID          : DINT
  REAL_val[0..9] : REAL
  ii               : DINT
  File_path        : String, len = 128, initial val = '\System_Disk\Real_data2.txt'
  File_Status      : String, len=128
 *)

 (* Set Read_File as TRUE to read data from the file *)
if  Read_File  then

   Read_File := FALSE ;
   (* Check if file exists *)
   if  F_exist( File_path ) = FALSE  then
      (* file doesn't exist *)
      File_Status := 'File "' + File_path +'" does not exist !' ;
   else

     (* file does exist , open it in read mode *)
      File_ID := F_Ropen( File_path );

      if  File_ID = 0  then
         (* open file in read mode fail *)
         File_status := 'Can not open File "' + File_path +'" !' ;
      else
```

Because the size of \System_Disk\ is small, recommend you amy change the directory to below (Depends on your application):

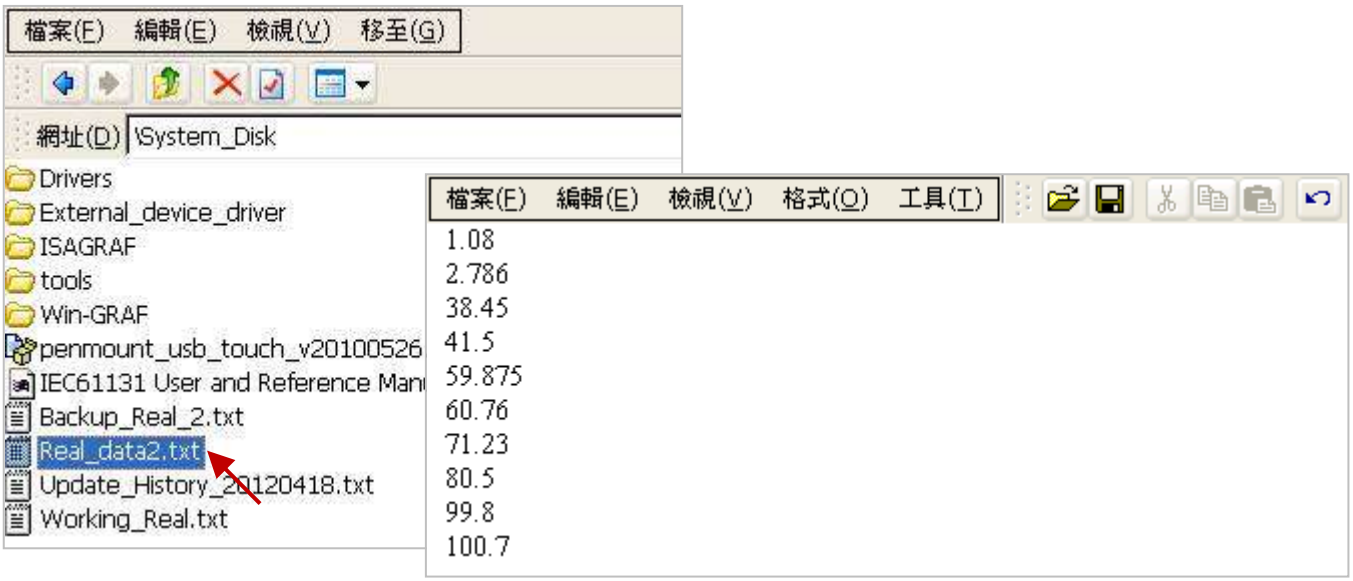
WinPAC, ViewPAC Series:
\Micro_SD or

XPAC Series:
\System_Disk2

```
(* open file in read mode ok, read REAL[0] ~ [9] from file ,
   each row contains 1 REAL value and end with <CR><LF> *)
File_status := 'Open File "' + File_path +'" Ok .' ;
for ii := 0 to 9  by 1  do

  (*  test if the end of file is reached in a file open for read *)
  if  F_EOF( File_ID )  then
    (* reach the end of file, exit "for loop" *)
    exit ;
  end_if ;

  (* read one row in the file as a string *)
  Tmp_string := FM_READ( File_ID ) ;

  (* convert the string to become REAL value *)
  REAL_val[ii] := Any_to_REAL( Tmp_string ) ;

end_for ;

(* close the file *)
F_close( File_ID ) ;

  end_if ;
 end_if ;
end_if ;
```

If want to read Integer, use the code below:
DINT_val[ii] := Any_to_**DINT**(Tmp_string) ;
and declare Variable “DINT_val” as DINT and Dim. at least "10" for this example..

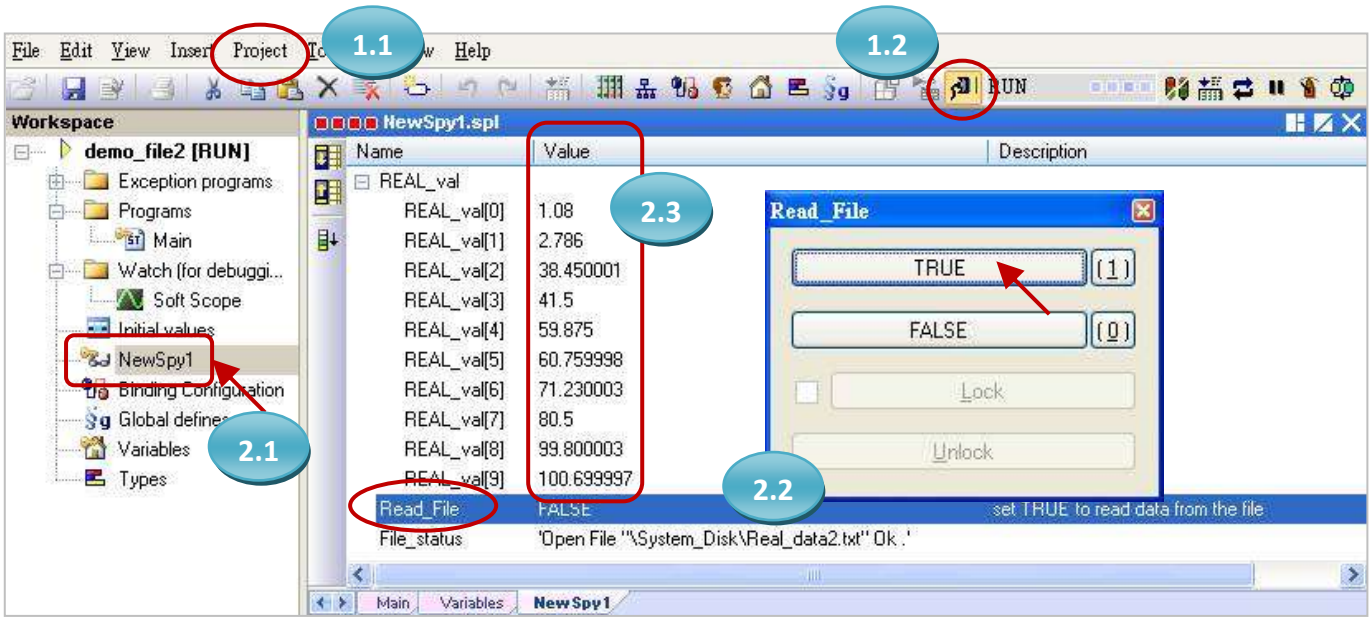
Note: In this example, when the "Read_File" is set to "TRUE", it will read the file on the PAC "\System_Disk\Real_data2.txt", please make sure the file already exists on the PAC.



Test Program:

1. Please set up IP configurations (Refer P12-1), compile and download the program to the PAC. (Click on "Project" > "Build All Projects" / "On Line", if not familiar with the operation, refer to Section 2.3.4 , Section 2.3.5)

2. Click "NewSpy1" to open a Spy List and set the "Read_File" to "TRUE" to read the data. (If OK, "File_Status" will show "Open File "\System_Disk\Real_data2.txt" Ok.")



Note: There is one another file operation example listed in the Section 6.2. It handles many data in the file, you may refer it.

12.4.3 Data Logging

Refer P12-1 to open this project ("demo_DataLog.zip") that provides a simple function for data logging. This program creates a spy list that contains a String variable (write_date), an Integer variable (int_data) and a Real variable (float_data). These variable values will be recorded to a file per minute and be stored on the PAC's **\System_Disk2**. This log file will be named according to the current date (e.g., May 06, 2015, the file name will be "2015-5-6.csv"). Moreover, this log file will be replaced per day and the existing file will be moved to "\System_Disk2\Current month\" (e.g., May 06, 2015, the file will be moved to "\System_Disk2\2015-5\").

Note: **This example project is used for the XP-8xx8-CE6 PAC. If you want to use the others PAC, simply change the file path as \Micro_SD\ or other storage location.**

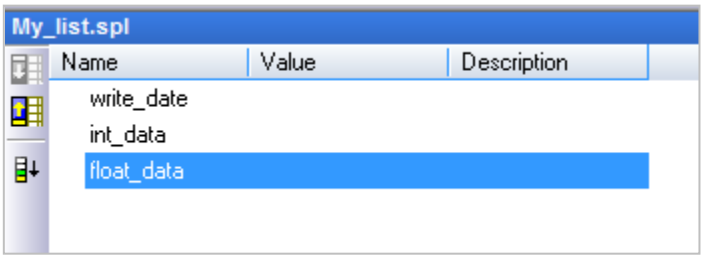
The sample of the CSV file format:

```
Time ,    int_val ,  float_val
20:18:30,  1236,    14.56
20:18:40,  3456,    34.56
20:18:50,  8932,    89.32
```

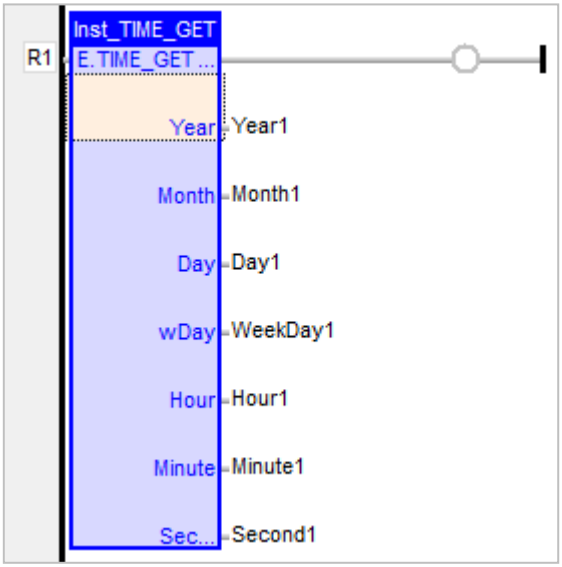
Description of variables: The user can view/set variables in the Win-GRAF "Variables" window.

Name	Data Type	Description
Year1	DINT	Used to get the PAC system time in the "PAC_Time" program.
Month1	DINT	
Day1	DINT	
WeekDay1	DINT	
Hour1	DINT	
Minute1	DINT	
Second1	DINT	
old_day	DINT	Used to know if the time is changed and then rename the log file in the "PAC_Time" program.
old_hour	DINT	
log_tmr1	TIME	Timer.
log_tmr2	TIME	
CSV_Path	STRING	The storage path of CSV file.
CSV_Dir	STRING	The storage folder of CSV file.
write_date	STRING	Used to record the time when writing data to a CSV file.
init	BOOL	Set it as TRUE to initialize. (Init value = TRUE)
int_data	DINT	Used to record the data.
float_data	REAL	Used to record the data.
writecsv	BOOL	Set it as TRUE to write data.
File_ID	DINT	Used for the "F_WOPEN" function.
tmp_bval	BOOL	The temporary variable.
tmp_msg	STRING	

Spy List – “My_List”:
There is a Spy list called “My_list” in this example project. (Refer Section 11.3 for more details)



LD Program – “PAC_Time”:
To get/set the PAC system time.



ST Program – “Main”:
To generate random values as the data of the log file (.csv) and to change the file storage path.

```
(* Declare “init” as "BOOL" ; Declare "log_tmr1" and "log_tmr2" as "TIME"  *)
(* Set “init” as TRUE to start ticking "TMR1"  *)
 IF    init    THEN
   init  :=  FALSE ;
      (* Enable the timer of the data logger *)
         TSTART (log_tmr1) ;
      (* Set the separator and decimal within the csv file  *)
        SetCsvOpt( ' , ' ,  ' . ' );
END_IF;

IF old_day<> day1 THEN
        tmp_msg:='\System_Disk2\'+CSV_Dir+'\'+ANY_TO_STRING(year1)+'-'+
                    ANY_TO_STRING(month1)+'-'+ANY_TO_STRING(old_day)+'.csv';
```

```
    (* Copy the file to \System_Disk2\ *)
        tmp_bval:=F_COPY (CSV_Path,tmp_msg);
    (* Delete the original file stored at \temp\ *)
        tmp_bval:=F_DELETE(CSV_Path);

    (* Create the CSV file and its filed name *)
        tmp_msg:='\Temp\'+any_to_string(year1)+'-'+any_to_string(month1)+'-'+
                    any_to_string(day1)+'.csv';
       CSV_Path:=tmp_msg;
       File_ID:= F_WOPEN(tmp_msg);

      IF File_ID<>0 THEN
           tmp_msg:='Time,Int_val,Float_val';
           tmp_bval:=FM_WRITE(File_ID,tmp_msg);
           tmp_bval:=F_CLOSE(File_ID);
       END_IF;

       old_day:=day1;
END_IF;

(* Create the folder every month, or when the PAC boot up. *)
if  old_month<>month1 then
       CSV_Dir:=ANY_TO_STRING(year1)+ANY_TO_STRING(month1);
       tmp_msg:='\System_Disk2\'+CSV_Dir;
       F_DIR(tmp_msg);
       old_month:=month1;
end_if;
```

Because the size of \System_Disk\ is small, we recommend that you save the log file at **\Micro_SD** (for WinPAC, ViewPAC series) or **\System_Disk2** (for XPAC series).

```
(* The function for writing data to a CSV file *)
IF  log_tmr1>=log_tmr2  THEN
    log_tmr1:=t#0s;

    (* Generate random values for the "Int_val" and the “Float_val" field *)
        int_data:=rand(1000);
        float_data:=ANY_TO_REAL(int_data)/100;
    (* The time for each data logging *)
       write_date:=ANY_TO_STRING(hour1)+':'+ ANY_TO_STRING (Minute1)+':'+
                    ANY_TO_STRING (Second1);
    (* Delete the previous file *)
       tmp_msg:='\System_Disk2\'+CSV_Dir+'\'+ANY_TO_STRING(Year1) +'-'+
                    ANY_TO_STRING(month1)+'-'+ANY_TO_STRING(Day1)+'.csv';
```

```
    (* Delete the file stored at \System_Disk2\ *)
        tmp_bval:=F_DELETE(tmp_msg);
    (* Trigging it to write data to a file *)
        writecsv:=true;
  END_IF;
```

LD Program – "WriteFile"

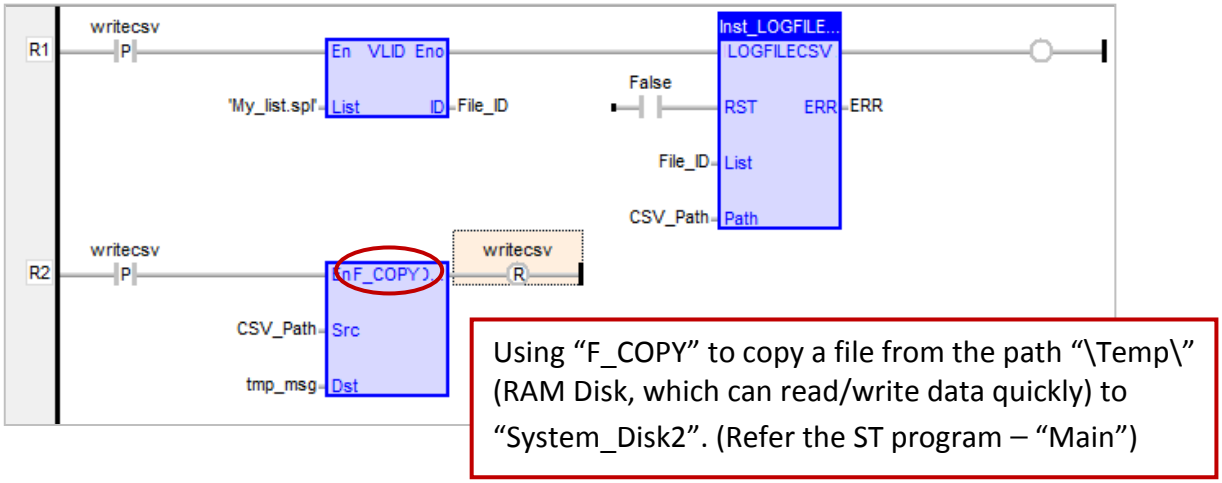
Set the "writecsv" as "TRUE" to write one data to the CSV file.

The user can click the following function or function block, and then press "F1" to view the descriptions.

VLID: Get the identifier of an embedded list of variables (i.e., Spy List - My_list.spl).

LogFileCSV: Generate a log file in CSV format for a list of variables.

F_COPY: Copy a file.



Chapter 13 VB.net 2008 Program Running in WP-8xx8 Access to Win-GRAF Variables

This chapter lists the procedure for creating the first demo program by Visual Studio .NET 2008 development tool. There are some sample programs in the WP-8xx8 CD-ROM.

VB .NET example:

CD-ROM : \napdos\Win-GRAF\demo-project\vb.net_2008_demo\

demo_vb01 : Digital I/O demo with one I-87055W in slot 0 of the WP-8xx8.

demo_vb02 : Analog I/O demo with one I-87024W in slot 1, one I-8017HW in slot 2.

demo_vb03 : Read/Write Win-GRAF internal integers, timers & real variables. (No I/O)

demo_vb04 : Read/Write Win-GRAF internal String variables. (No I/O)

Win-GRAF example:

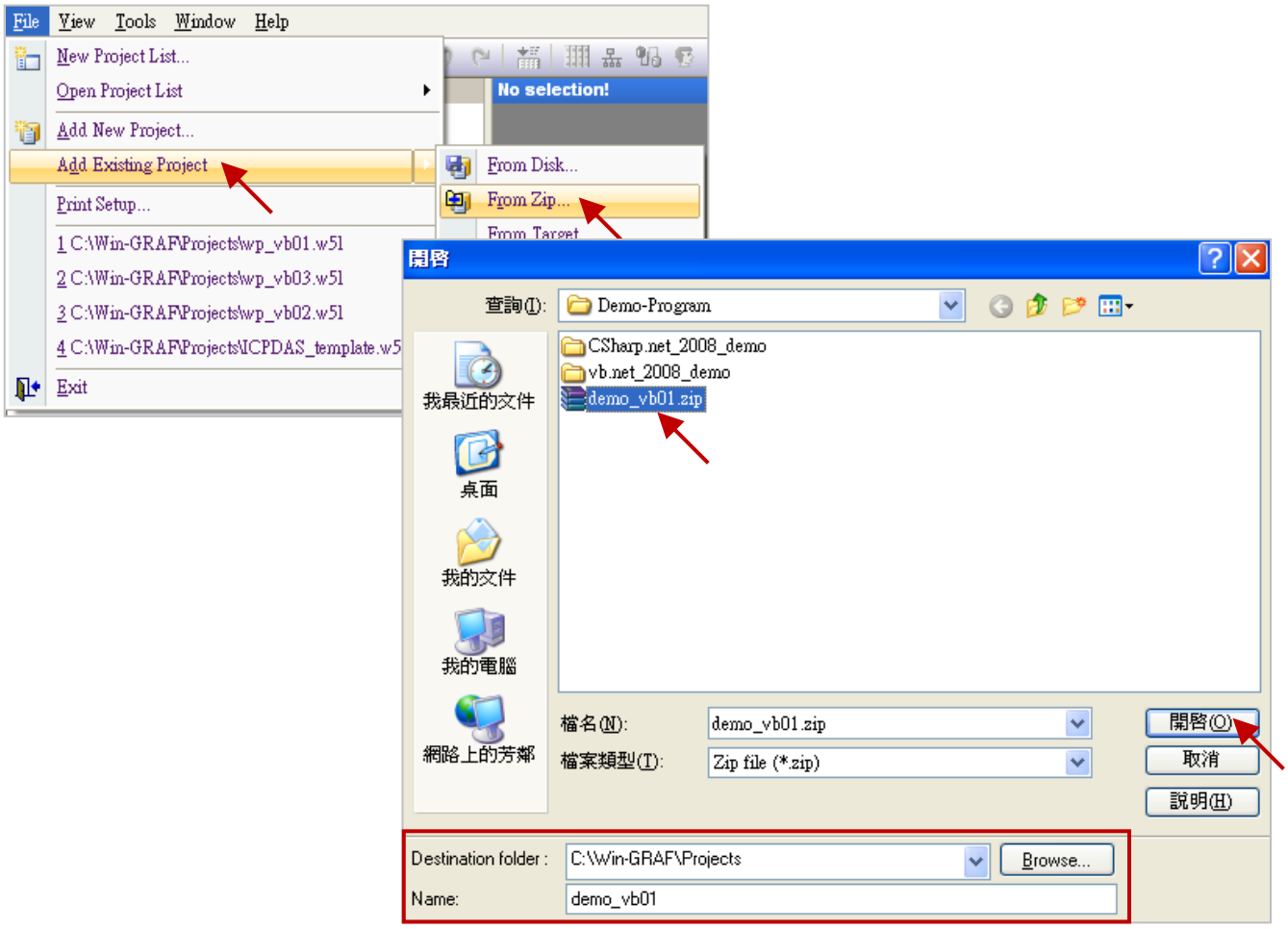
CD-ROM : \napdos\Win-GRAF\demo-project\

"demo_vb01.zip", "demo_vb02.zip", "demo_vb03.zip", "demo_vb04.zip"

13.1 Add an Existing Win-GRAF Project from a ZIP

Please follow these steps to restore the Win-GRAF project.

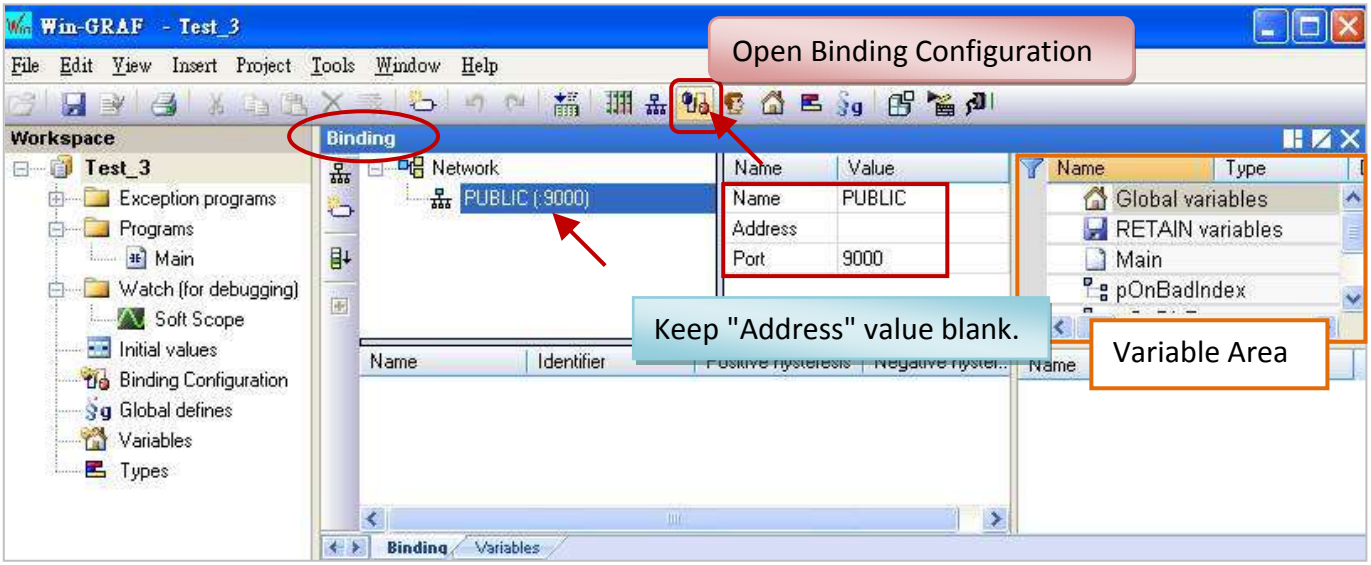
First Click “File” -> “Add Existing Project” -> “FromZip…”. Then choose the Win-GRAF project zip file which you would like to restore. After restoring the project, you have to build the project, and then download it to the PAC.



13.2 Publishing the Win-GRAF Variable for .NET and Soft-GRAF HMI

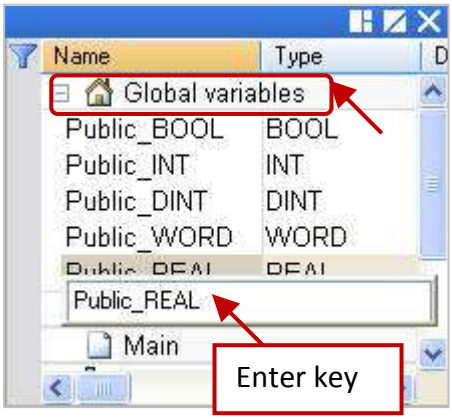
If users wish to use .NET program to Read/Write the Win-GRAF variables. Except for String variable all of the variables need to use the "Open Binding Configuration" function to set an address. The following demonstrates how to publish Win-GRAF variable:

1. Click “Open Binding Configuration” on the toolbar to open the “Binding” setup window.
2. Click "PUBLIC (:9000)". Keep the "Address" value is blank and "Port" value is fixed to 9000.



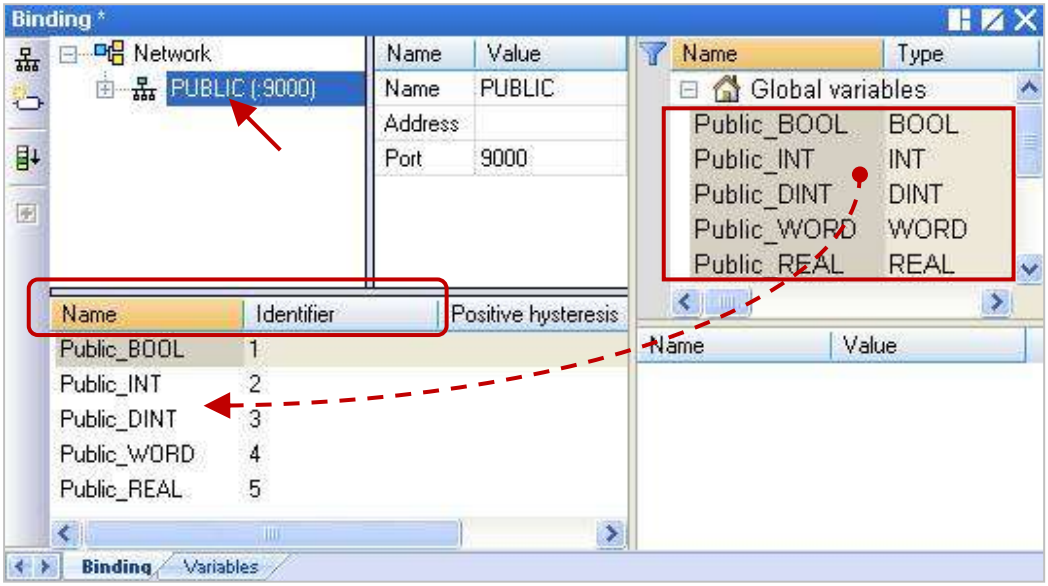
3. Before publishing these variables, make sure you have declared them in the Variables Area. Click “Global variables” and press the “Ins” key to insert a new variable. The following table demonstrates variables using in the “Test_3” project and you can declare them according to the needs of your application.

Variable name	Type
Public_BOOL	BOOL
Public_INT	INT
Public_DINT	DINT
Public_WORD	WORD
Public_REAL	REAL



4. As the figure below, click on "PUBLIC(:9000)" and drag all the needed variables to the "Name" area. The "Identifier" will generate an address number automatically. If any other VB or .NET program wants to use these public variables, it must set to the same address number (ID).

NOTE:
The "PUBLIC" allows to use up to 8192 variables, and the "Identifier" number JUST can be 1 to 8192.



The following procedure will show you how to use the "pub_string" function to publish the Win-GRAF String variable in the ST program.

Syntax:

```
Pub_string(Address, String_val) ;
```

Address: The public address number, and its range can be 1 to 1024
String_val: The name of String variable.

Variables description:

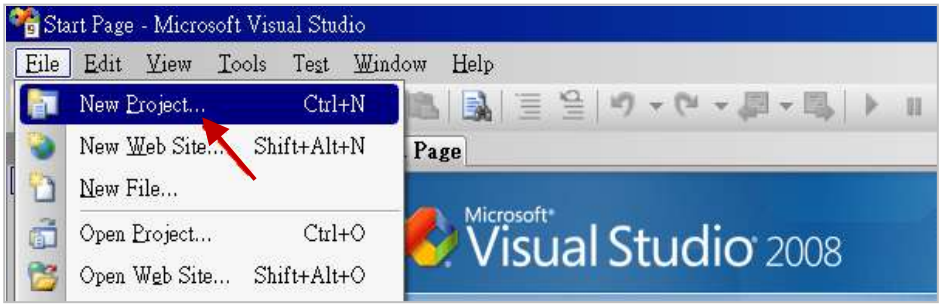
Name	Type	Description
Init	BOOL	Set the initial value as TRUE.
Tmp_val	BOOL	TRUE: Binding succeeds. FALSE: Binding fails.
msg1	STRING, Length is 100	String variable for demo purpose. **NOTE: The String length could be 1 to 255.**
msg2	STRING, Length is 32	
msg3	STRING, Length is 60	

ST program:

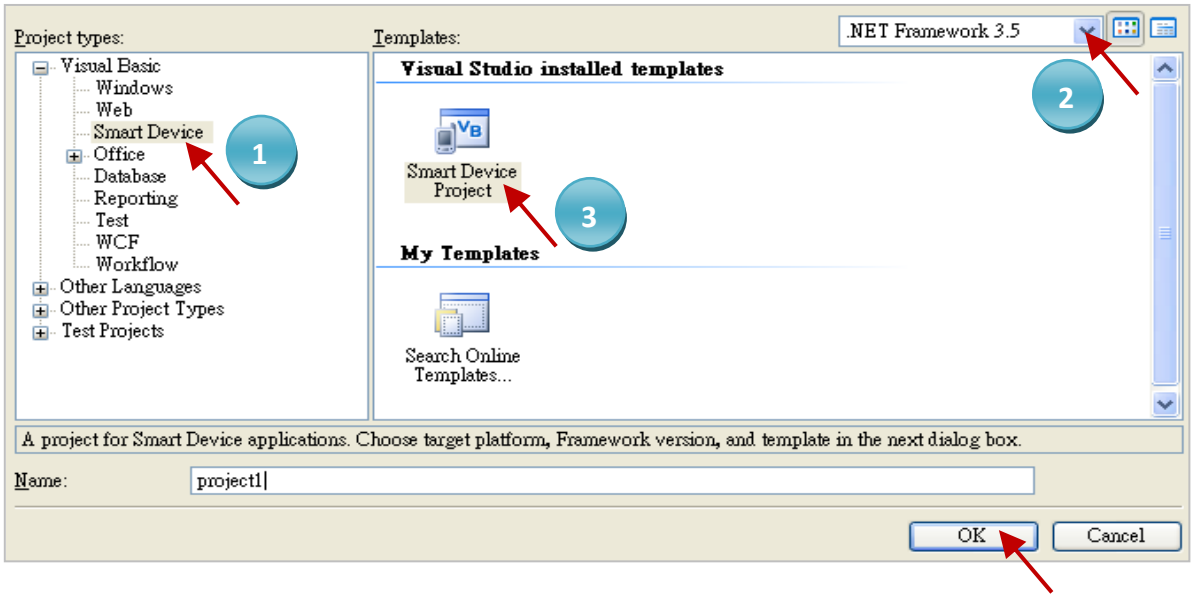
```
If  init  then
  Init := false;
(*add address 1 for share string val *)
  Tmp_val := pub_string(1,msg1);

(*add address 2 for share string val *)
  Tmp_val := pub_string(2,msg2);

(*add address 3 for share string val *)
  Tmp_val := pub_string(3,msg3);

End_if;
```

13.3 Create a new VB.NET project

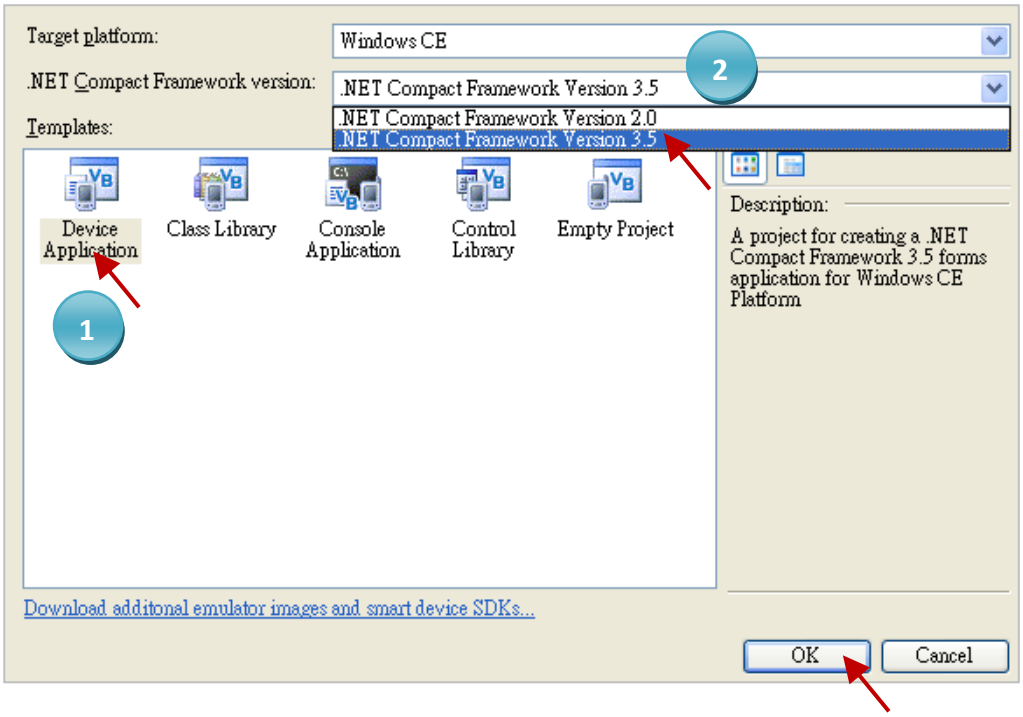
1. First, run Microsoft Visual Studio .NET 2008 software, and then choose “File” > “New Project”.



2. Click “Smart Device” on the left, and then select “.NET Framework 3.5” and “Smart Device Project”. Entering a proper project name and click “OK”.



3. Select the "Device Application" and "Windows CE" and “.NET Compact Framework Version 3.5”, then click “OK”.

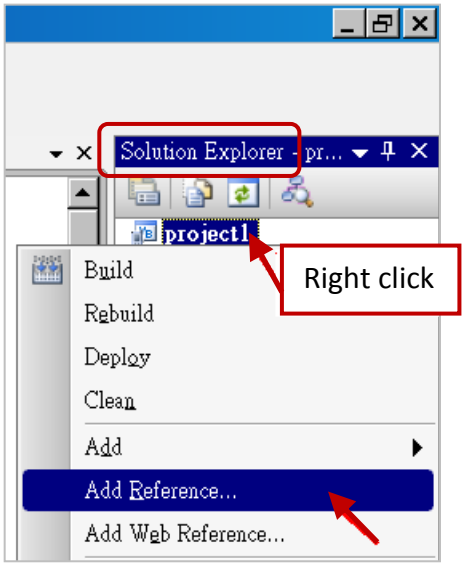


13.3.1 Add Project Reference

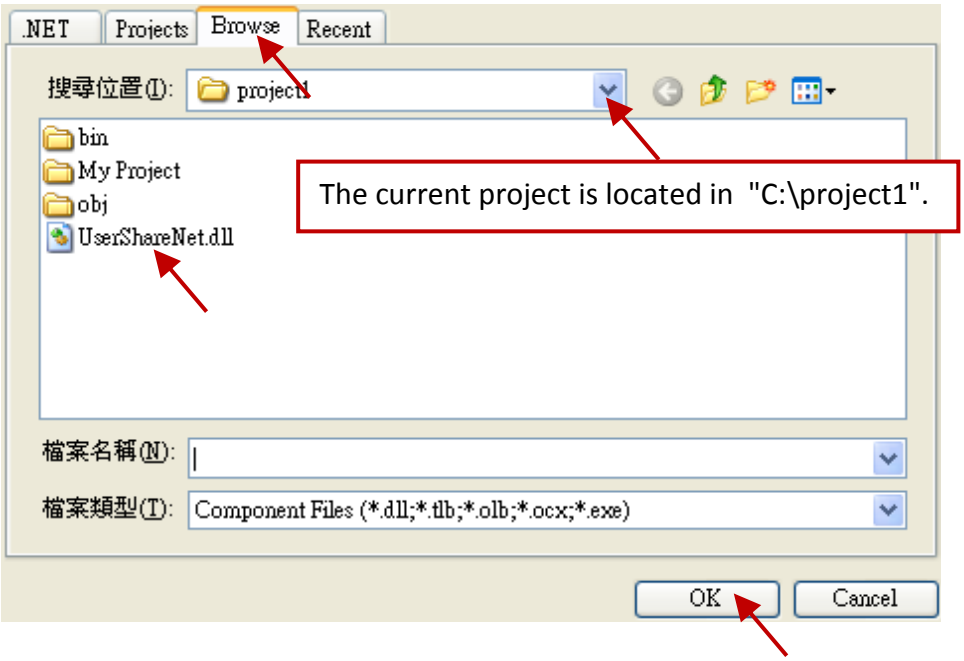
The “UserShareNet.dll” library contains all functions of data exchange with Win-GRAF variables. Before you use the “UserShareNet” keyword in the program, you must add the “UserShareNet.dll” into the reference list of your project.

1. Copy the “UserShareNet.DLL” from Win-PAC’s shipment CD (\napdos\Win-GRAF\WP-8xx8\vb.net_2008_demo\wp_vb01\vb01\) to your project folder (e.g., “C:\project1\”)

2. Right click on the project name (e.g., “project1”) in the “Solution Explorer” window, and then select “Add Reference …”.

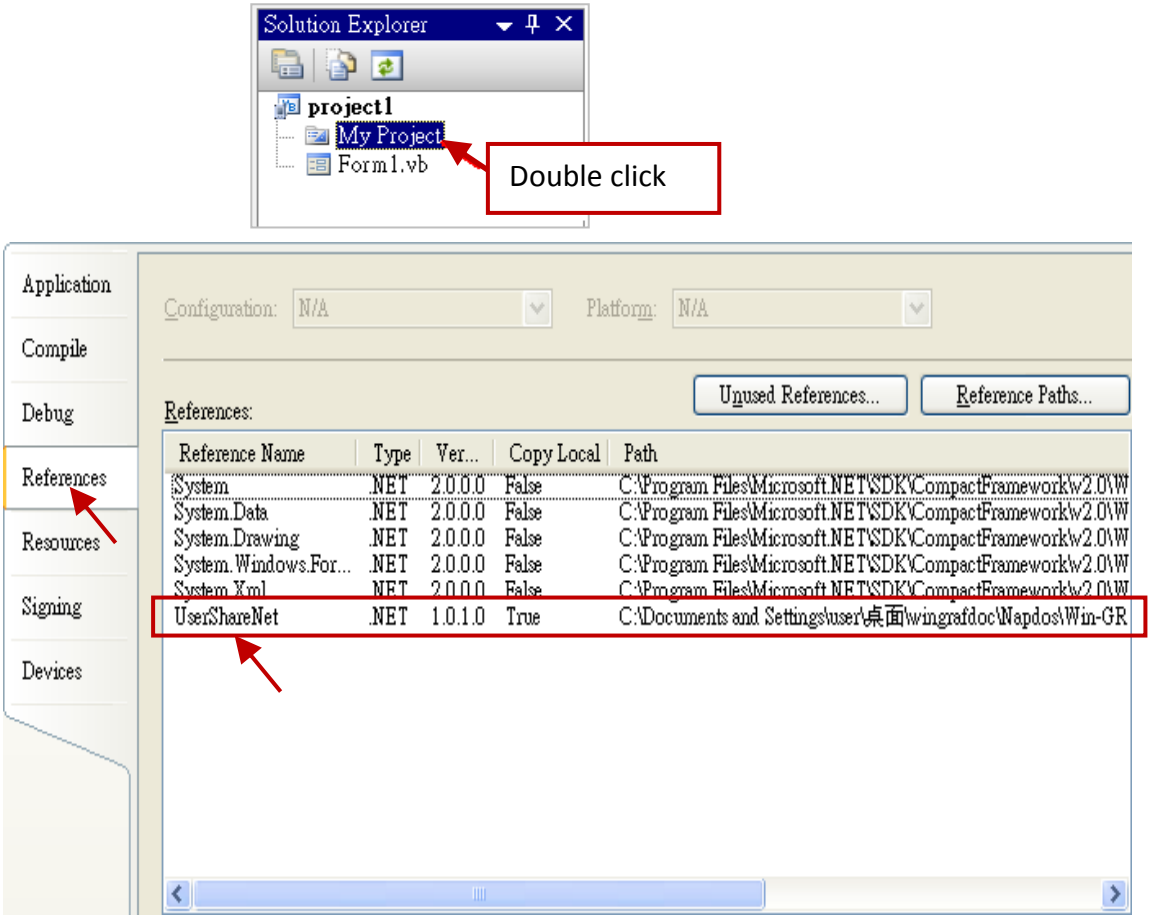


3. Click the “Browse” tab and select the “UserShareNet.dll” from your project location.

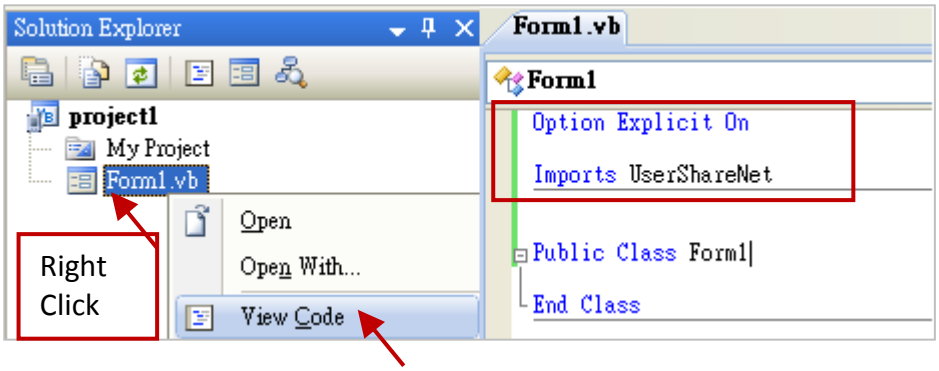


Note: You may copy the “UserShareNet.dll” from the CD-ROM to your current project path first. Then add it to the project reference.

4. When “UserShareNet.dll” is added, please double click on “My Project” to check if the “UserShareNet.dll” is well added.



5. Right-click on the “Form1.vb” and select “View Code” from the pop-up. Move cursor to top and insert the “Option Explicit On” and “Imports UserShareNet” in the first two statements.

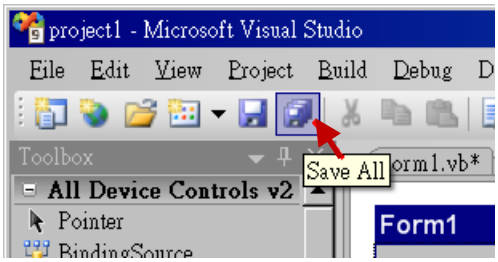


Then you can design all required objects and actions inside your VB Forms.
(Refer the Chapter 13.5 for more information about using functions in the “UserShareNet.dll”.)

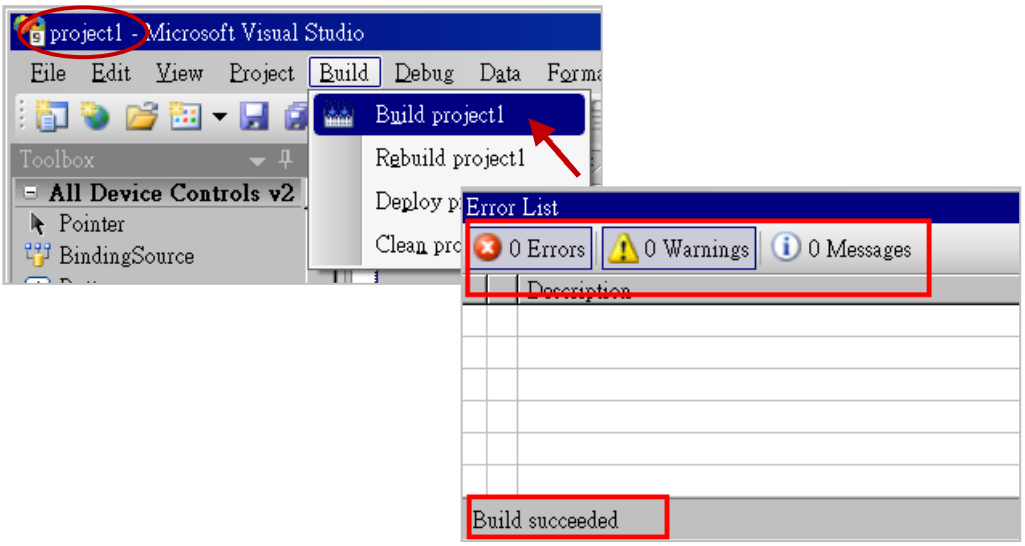
13.4 Compiling the Application

When you have finished writing a program, you can build (compile) an application by the following steps.

1. Remember to save at any time for safety.



2. Then compile (Build) the project. The result is listed in the "Error List" windows at the bottom.



3. You can find the execution file in

<Your VB.net Project folder> \bin\Release\ **<project_name>.exe**

Please copy this execution file to the WP-8xx8's \System_Disk\Win-GRAF\ path to run it.

Note:

The user may copy the VB.net execution file to another path to run it, but there should contain at least two DLL files with it or it cannot run correctly.

For ex, the project1.exe can run in the \Micro_SD\ folder if there are three files in it. The "project1.exe" , "UserShareNet.dll" and "Quicker.dll" . (The "UserShareNet.dll" and "Quicker.dll" can be copied from the Win-GRAF PAC's "\System_disk\Win-GRAF\" path)

13.5 UserShareNet.dll

This section we will focus on the description of the application example of UserShareNet.dll functions. There are some functions that can be used to read/write data from/to the Win-GRAF soft-logic. The functions of UserShareNet.dll can be divided into as listed below:

1. R/W Boolean
2. R/W 8-bit Integer
3. R/W 16-bit Integer
4. R/W 32-bit Integer
5. R/W 64-bit Integer
6. R/W 32-bit Float
7. R/W 64-bit Float
8. R/W String

※ **Refer the "Appendix A" to get familiar with the definition of Win-GRAF variables.**

13.5.1 R/W Boolean Functions

Set_BOOL

Description:

This function is to set a value to a Win-GRAF Boolean variable.

Syntax:

```
UserShare.Set_BOOL ( iUserAddress  As System.UInt16 , ByVal iStatus As byte) as Byte
```

Parameter:

iUserAddress : Address of the Variable (1 to 8192)

iStatus : Set the status. For instance, iStatus = 1 for True, iStatus = 0 for False

Example:

'Set the Win-GRAF BOOL variable with address 1 to True.

UserShare.Set_BOOL(Convert.ToUInt16(1), 1)

Demo program:

CD-ROM: \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb01

Get_BOOL

Description:

This function is to get the value from a Win-GRAF BOOL variable.

Syntax:

```
UserShare.Get_BOOL ( iUserAddress As System.UInt16 , ByRef iStatus As byte)
```

Parameter:

iUserAddress : Address of the Variable (1 to 8192)

iStatus : Get the variable value , iStatus = 1 for True, iStatus = 0 for False

Example:

```
'Get the value of Win-GRAF BOOL variable with address 1.
Dim iStatus As Byte
UserShare.Get_BOOL(Convert.ToUInt16(1), iStatus)
```

Demo Program:

CD-ROM: \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb01

13.5.2 Integer R/W Functions

■ **Set_SINT** ■ **Set_INT** ■ **Set_DINT** ■ **Set_LINT**

Description:
These functions are to set 8-bit Integer, 16-bit Integer, 32-bit integer & 64-bit Integer value to Win-GRAF integer variables.

Syntax:

UserShare.Set_SINT (ByVal *iUserAddress* As System.UInt16 , ByVal *iStatus* As SByte) As Byte

UserShare.Set_INT (ByVal *iUserAddress* As System.UInt16 , ByVal *iStatus* As Short) As Byte

UserShare.Set_DINT (ByVal *iUserAddress* As System.UInt16 , ByVal *iStatus* As Integer) As Byte

UserShare.Set_LINT (ByVal *iUserAddress* As System.UInt16 , ByVal *iStatus* As long) As Byte

Parameter:
iUserAddress : Address of Variable. (1 to 8192)
iStatus : the value of 8-bit Integer, 16-bit Integer, 32-bit Integer or 64-bit Integer.

Example:
'Set a 32-bit integer value "1234567" to the Win-GRAF DINT variable with address "1".
UserShare.Set_DINT(Convert.ToUInt16(1), Convert.ToInt32(1234567))

'Set a 16-bit integer value "-1234" to the Win-GRAF INT variable with address "2".
UserShare.Set_INT(Convert.ToUInt16(3), Convert.ToInt16(-1234))

'Set a 64-bit Integer value "123456789012345" to the Win-GRAF LINT variable with address "3".
UserShare.Set_LINT(Convert.ToUInt16(3), Convert.ToInt64(123456789012345))

'Set a 8-bit Integer value "125" to the Win-GRAF SINT variable with address "4".
UserShare.Set_SINT(Convert.ToUInt16(3), Convert.ToSByte(125))

Demo Program:
CD-ROM:
1. \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb02 for R/W analog I/O
2. \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb03 for R/W internal long integer, Timer and Real (floating-point) values.

■ Get_SINT ■ Get_INT ■ Get_DINT ■ Get_LINT

Description:

These functions are to get 8-bit integer, 16-bit integer, 32-bit integer & 64-bit integer value from Win-GRAF integer variables.

Syntax:

UserShare. Get_SINT (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As SByte) As Byte

UserShare. Get_INT (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As Short) As Byte

UserShare.Get_DINT (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As Integer) As Byte

UserShare. Get_LINT (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As long) As Byte

Parameter:

iUserAddress : Address of Variable (1 to 8192)

iStatus : Get the 8-bit integer, 16-bit integer, 32bit-integer or 64-bit integer value.

Example:

```
Dim Dlong_val As Int64
Dim short_val As Int16
Dim long_val As Int32
Dim Sbyte_val as byte

'Get 64-bit integer value from the Win-GRAF LINT variable with address "7".
UserShare.Get_LINT(Convert.ToUInt16(7), Dlong_val)

'Get 32-bit integer value from the Win-GRAF DINT variable with address "8".
UserShare.Get_DINT(Convert.ToUInt16(8), long_val)

'Get 16-bit integer value from the Win-GRAF INT variable with address "9".
UserShare.Get_INT(Convert.ToUInt16(9), short_val)

'Get 8-bit integer value from the Win-GRAF SINT variable with address "10".
UserShare.Get_SINT(Convert.ToUInt16(9), sbyte_val)
```

Demo program:

CD-ROM:

1. R/W analog I/O :
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo \demo_vb02
2. R/W internal long integer, Timer and Real (floating-point) values :
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb03

13.5.3 R/W Real Variable Functions

■ Get_REAL ■ Get_LREAL

Description:
These functions are to get 32-bit REAL and 64-bit double from the Win-GRAF REAL/LREAL variable.

Syntax:

UserShare. Get_REAL (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As Single) As Byte

UserShare. Get_LREAL(ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As Double) As Byte

Parameter:
iUserAddress : Address of Variable (1 to 8192)
iStatus : Get the 32-bit REAL or 64-bit double value.

Example:

```
Dim float_val As Single
Dim double_val As Double

'Get 64-bit double value from the Win-GRAF LREAL variable with address "7".
UserShare.Get_LREAL(Convert.ToUInt16(7), double_val)

'Get 32-bit REAL value from the Win-GRAF REAL variable with address "8".
UserShare.Get_REAL(Convert.ToUInt16(8), float_val)
```

Demo program:
CD-ROM:
1. R/W analog I/O:
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb02
2. R/W internal long integer, Timer and Real (floating-point) values :
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb03

■ Set_REAL ■ Set_LREAL

Description:

These functions are to set 32-bit REAL and 64-bit double value to the Win-GRAF REAL/LREAL variable.

Syntax:

UserShare. Set_REAL (ByVal *iUserAddress* As System.UInt16, ByVal *iStatus* As Single) As Byte

UserShare. Set_LREAL(ByVal *iUserAddress* As System.UInt16, ByVal *iStatus* As Double) As Byte

Parameter:

iUserAddress : Address of Variable. (1 to 8192)
iStatus : Set the 32-bit REAL or 64-bit double.

Example:

'Set a 64-bit double value "11234.234567" to the Win-GRAF LREAL variable with address "1".
UserShare.Set_LREAL(Convert.ToUInt16(7),Convert.ToDouble(11234.234567))

'Set a 32-bit REAL value "123.12" to the Win-GRAF REAL variable with address "8".
UserShare.Set_REAL(Convert.ToUInt16(8), Convert.ToSingle (123.12))

Demo program:

CD-ROM:

1. R/W analog I/O : \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb02
2. R/W internal long integer, Timer and Real (floating-point) values :
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb03

13.5.4 R/W String Variable Functions

Get_STRING

Description:
This function is to get a Win-GRAF String variable.

Syntax:

UserShare. Get_STRING (ByVal *iUserAddress* As System.UInt16, ByVal *msg()* As Byte) As Byte

Parameter:
iUserAddress : Address of Variable (1 to 1024)
msg() : Get the string value.

Example:

```
Dim str_val As String
Dim msg() As Byte

'Get String value of the Win-GRAF String variable with address "7".
UserShare.Get_STRING(Convert.ToUInt16(7),msg )
str_val= byte_array_to_unicode(msg)

 Private Function byte_array_to_unicode(ByVal buf() As Byte) As String
     Dim tmpmsg As String
     If buf.Length > 255 Then
        Return Nothing
     End If
tmpmsg = System.Text.Encoding.GetEncoding("UTF-8").GetString(buf, 0,  buf.Length)

     Return tmpmsg
  End Function
```

Demo program:
CD-ROM:
1. R/W String variable:
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb04

Set_STRING

Description:

This function is to set a String value to the Win-GRAF String variable.

Syntax:

UserShare. Set_STRING (ByVal *iUserAddress* As System.UInt16, ByVal *msg()* As Byte) As Byte

Parameter:

iUserAddress : Address of Variable. (1 to 1024)

msg() : the string value.

Example:

```
Dim str_val As String=”Hello World”
Dim msg() As Byte

msg= unicode_to_byte_array(str_val)

‘Set a string value “Hello World” to the Win-GRAF String variable with address “7”.
UserShare.Set_STRING(Convert.ToUInt16(7),msg )

‘Convert String to byte array.
Private Function unicode_to_byte_array(ByVal msg As String) As Byte()
    Dim tmpbuf() As Byte
    If msg.Length > 255 Then
      Return Nothing
    End If
    tmpbuf = System.Text.Encoding.GetEncoding("UTF-8").GetBytes(msg)
    Return tmpbuf
End_Function
```

Demo program:

CD-ROM:

1. R/W String variable:
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb04

13.5.5 How to use VB.NET R/W to Win-GRAF String Variable

Before .NET program write to Win-GRAF String variable. The String-type has to convert to byte array. If you need to read Win-GRAF String variable. Then you have to convert byte array to String. There is a VB.NET example to show how to convert each other.

(Encode :UTF-8)：

Convert String to byte array

```
Private Function unicode_to_byte_array(ByVal msg As String) As Byte()
    Dim tmpbuf() As Byte
    If msg.Length > 255 Then
        Return Nothing
    End If

    tmpbuf = System.Text.Encoding.GetEncoding("UTF-8").GetBytes(msg)
    Return tmpbuf
End Function
```

Convert byte array to string

```
Private Function byte_array_to_unicode(ByVal buf() As Byte) As String
    Dim tmpmsg As String
    If buf.Length > 255 Then
        Return Nothing
    End If
tmpmsg = System.Text.Encoding.GetEncoding("UTF-8").GetString(buf, 0, buf.Length)

    Return tmpmsg
End Function
```

Chapter 14 C# .net 2008 Program Running in WP-8xx8 Access to Win-GRAF Variables

This chapter lists the procedure for creating the first demo program by Visual Studio .NET 2008 development tool. There are some sample programs in the WP-8xx8 CD-ROM.

C# demo:

CD-ROM : \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\
demo_CSharp01 : Digital I/O demo with one I-87055W in slot 0 of the WP-8xx8.
demo_CSharp02 : Analog I/O demo with one I-87024W in slot 1 and one I-8017HW in slot 2.
demo_CSharp03 : Read / Write Win-GRAF internal integers, timers and real variables. (No I/O)
demo_CSharp04 : Read/Write Win-GRAF internal String variables. (No I/O)

Win-GRAF demo:

CD-ROM : \napdos\Win-GRAF\demo-project\
"demo_vb01.zip", "demo_vb02.zip", "demo_vb03.zip", "demo_vb04.zip"

14.1 Add an Existing Win-GRAF Project from a ZIP

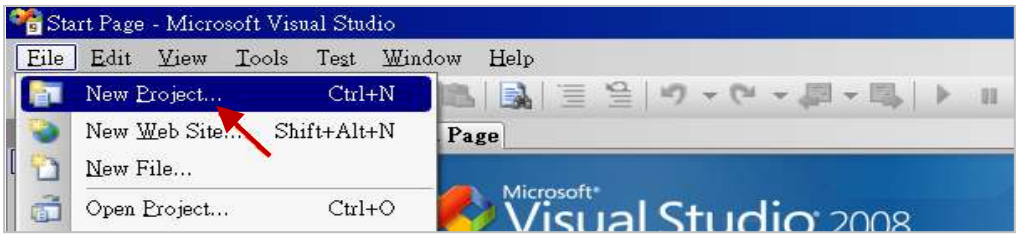
Please refer the Chapter 13.1

14.2 Publishing the Win-GRAF Variable for .NET

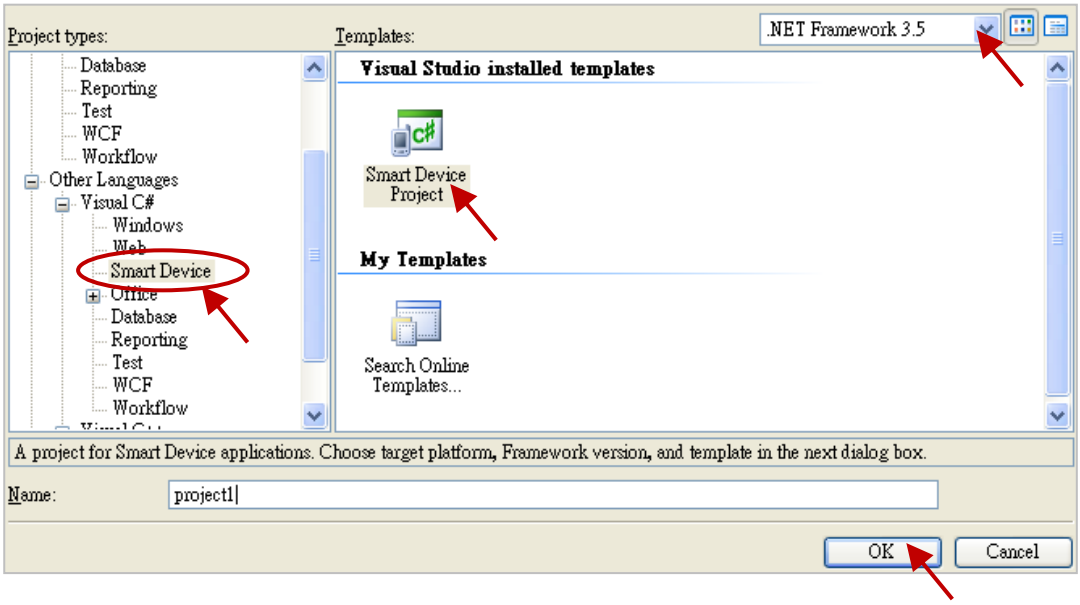
Please refer the Chapter 13.2

14.3 Create a New C# Project

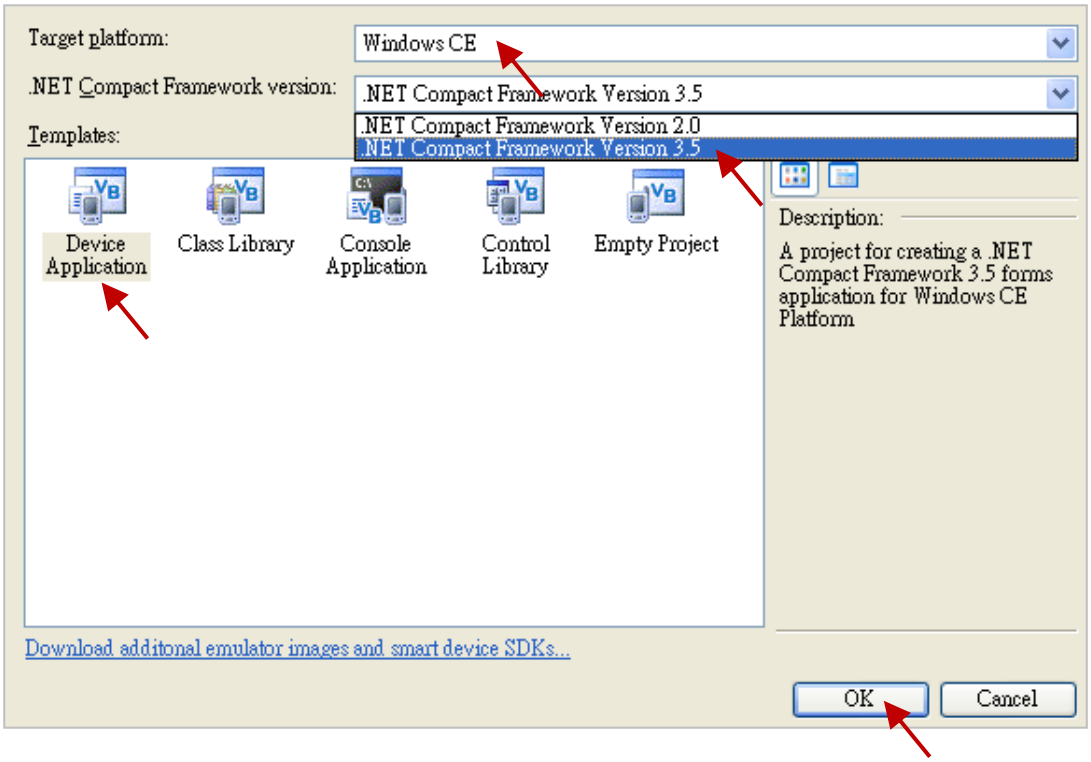
1. First, users need to open Microsoft Visual Studio .NET 2008 software. And then in the menu of “File”, please run the “New Project” .



2. Check the “Smart Device” on the left, then selecting the “.NET framework 3.5” and “Smart Device Project”. Then entering a proper project name and the last click on “OK”.



3. Select the "**Device Application**" and "**Windows CE**" and “**.NET Compact Framework Version 3.5**”, then click on “OK”.

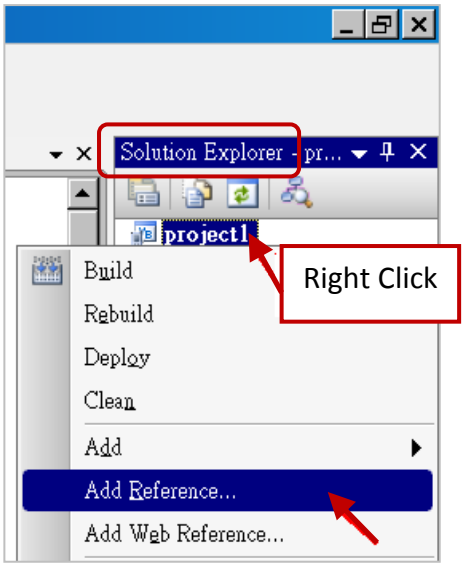


14.3.1 Add C# Project Reference

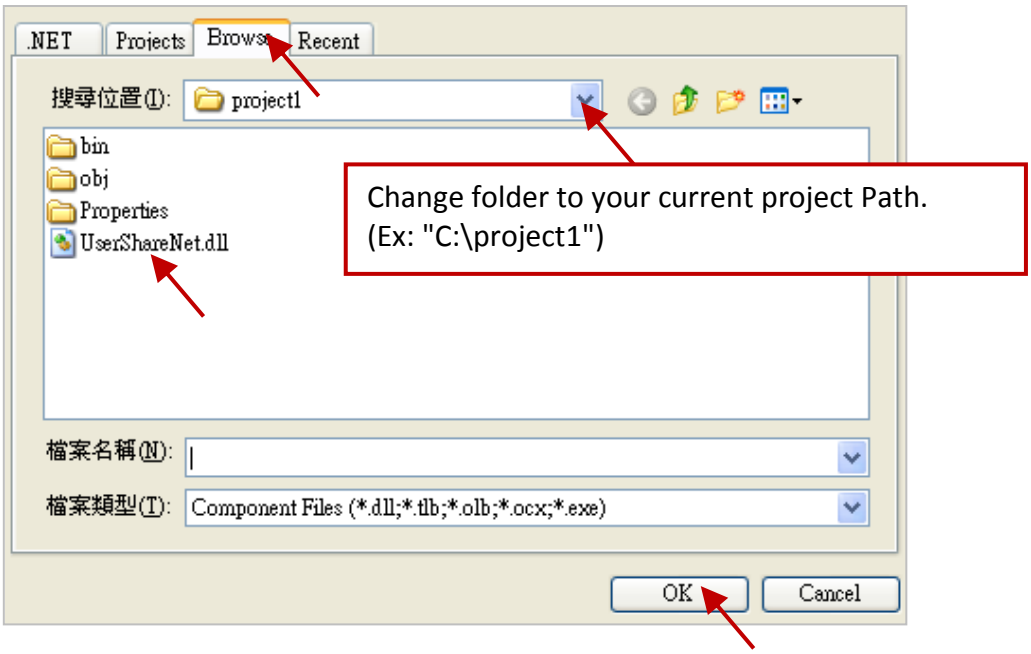
The "UserShareNet" library contains all modules' functions. Before you use the "UserShare" keyword in the program, you must add the "UserShareNet.dll" into the reference list of your application.

1. Copy the "UserShareNet.DLL" from WP-8xx8 CD-ROM: **\napdos\Win-GRAF\WP-8xx8\CSharp.net_2008_demo\demo_CSharp01** to your project folder(ex: C:\project1\)

2. Right click on the Project name on the right hand side , then select "Add Reference …"

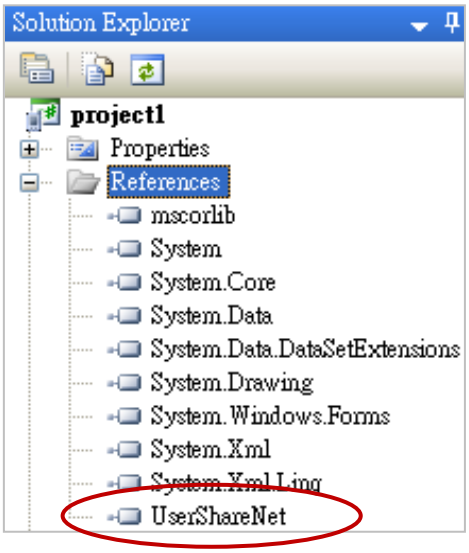


3. Click the "**Browse**" button. Select the "UserShareNet.dll" from your project location.

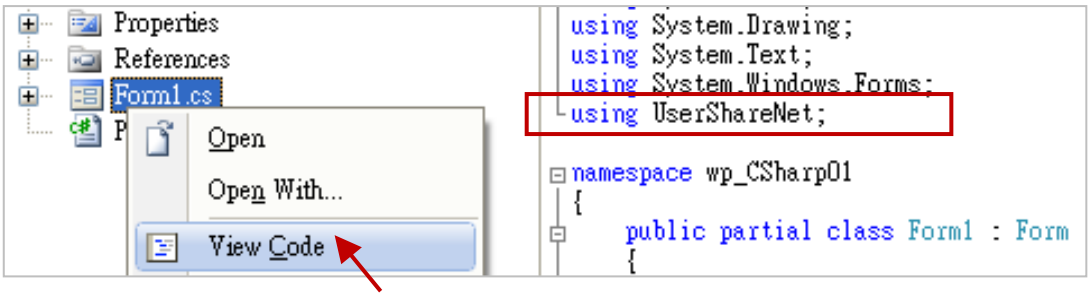


Note: **You may copy the "UserShareNet.dll" from the CD-ROM to your current project path first. Then add it to the project reference.**

4. When “UserShareNet.dll” are added, you can see them in the solution explorer as below.



5. Right-click on the “**Form1.cs**” and select “**View Code**” from the pop-up. Move cursor to top and insert the “**using UserShareNet;**” in the first statements.

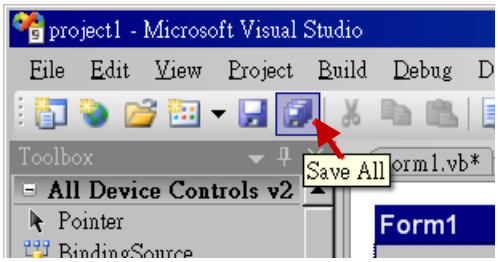


Then you can design all required objects and actions inside your C# Forms.
(Refer the Section 14.5 for more information about using functions in the “UserShareNet.dll”.)

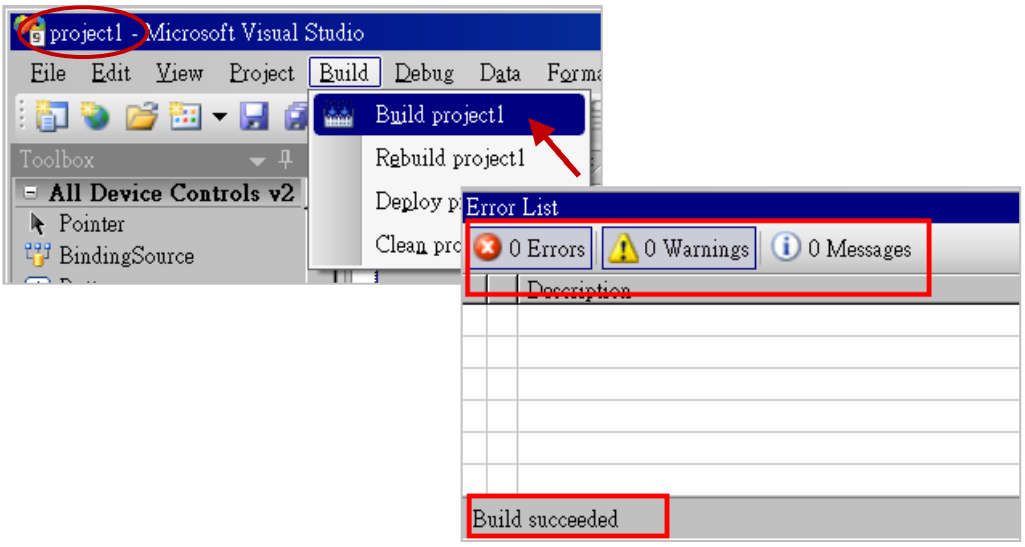
14.4 Compiling the Application Program

When you have finished writing a program, you can build (compile) an application by the following steps.

1. Remember to save at any time for safety.



2. Then compile (Build) the project . The result is listed in the "Error List" windows at the bottom.



3. You can find the execution file in

<Your C# .net Project folder> \bin\Release\ <project_name>.exe

Please copy this execution file to the WP-8xx8's \System_Disk\Win-GRAF\ path to run it.

Note:

The user may copy the C#.net execution file to another path to run it, but there should contain at least two DLL files with it or it cannot run correctly. For ex, the project1.exe can run in the \Micro_SD\ path if there are three files in it. The "project1.exe", "UserShareNet.dll" and, "Quicker.dll" .
(The "UserShareNet.dll" and "Quicker.dll" can be copied from the Win-GRAF PAC's "\System_disk\Win-GRAF\" path)

14.5 UserShareNet.DLL

This section we will focus on the description of the application example of UserShareNet.DLL functions. There are some functions that can be used to read/write data from/to the Win-GRAF variable. The functions of UserShareNet.DLL can be divided into as listed below

1. R/W Boolean
2. R/W 8-bit Integer
3. R/W 16-bit Integer
4. R/W 32-bit Integer
5. R/W 64-bit Integer
6. R/W 32-bit Float
7. R/W 64-bit Float
8. R/W 32-bit String

※ Refer the "Appendix A" to get familiar with the definition of Win-GRAF variables.

14.5.1 R/W Boolean Functions

■ Set_BOOL

Description:
This function is to set a value to a Win-GRAF Boolean variable.

Syntax:

```
UserShare.Set_BOOL(ushort iUserAddress, byte iStatus)
```

Parameter:
iUserAddress : Address of Variable (1 to 8192)
iStatus : Set the status. For instance, iStatus = 1 for True, iStatus = 0 for False

Example:

```
// Set the Win-GRAF BOOL variable with address 1 to True.
 UserShare.Set_BOOL(Convert.ToUInt16(1), 1);
```

Demo program:
CD-ROM : \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp01

Get_BOOL

Description:

This function is to get the value from a Win-GRAF BOOL variable.

Syntax:

```
UserShare.Get_BOOL(ushort iUserAddress, out byte iStatus)
```

Parameter:

iUserAddress : Address of Variable. (1 to 8191)

iStatus : Get the variable status , iStatus = 1 for True, iStatus = 0 for False.

Example:

```
Byte iStatus=0;
// Get the value of Win-GRAF BOOL variable with address 1.
UserShare.Get_BOOL(Convert.ToUInt16(1),out iStatus);
```

Demo program:

CD-ROM: \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp01

14.5.2 R/W Integer Functions

■ **Set_SINT** ■ **Set_INT** ■ **Set_DINT** ■ **Set_LINT**

Description:
These functions are to set 8-bit Integer, 16-bit Integer, 32-bit integer & 64-bit Integer value to Win-GRAF integer variables.

Syntax:

```
UserShare.Set_SINT(ushort iUserAddress , sbyte iStatus)

UserShare.Set_INT(ushort iUserAddress , short iStatus)

UserShare.Set_DINT(ushort iUserAddress, int iStatus)

UserShare.Set_LINT(ushort iUserAddress, long iStatus)
```

Parameter:
iUserAddress : Address of Variable. (1 to 8192)
iStatus : Set the 8-bit Integer, 16-bit Integer, 32-bit Integer or 64-bit Integer.

Example:

```
// Set a 32-bit integer value "1234567" to the Win-GRAF DINT variable with address "1".
int temp1=1234567;
UserShare.Set_DINT(Convert.ToUInt16(1), temp );

// Set a 16-bit integer value "-1234" to the Win-GRAF INT variable with address "2".
short temp2= -1234;
UserShare.Set_INT(Convert.ToUInt16(2), temp2 );

// Set a 64-bit Integer value "123456789012345" to the Win-GRAF LINT variable with address "3".
long temp3=123456789012345;
UserShare.Set_LINT(Convert.ToUInt16(3),  temp3 );

// Set a 8-bit Integer value "125" to the Win-GRAF SINT variable with address "4".
Sbyte temp4=125;
UserShare.Set_SINT(Convert.ToUInt16(4),  temp4 );
```

Demo program:
CD-ROM:
1. R/W analog I/O:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp02
2. R/W internal Boolean ,long integer, Timer and Real (floating-point) values :
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp03

■ Get_SINT ■ Get_INT ■ Get_DINT ■ Get_LINT

Description:

These functions are to get 8-bit integer, 16-bit integer, 32-bit integer & 64-bit integer value from Win-GRAF integer variables.

Syntax:

```
UserShare.Get_SINT(ushort iUserAddress, out sbyte iStatus)

UserShare.Get_INT(ushort iUserAddress, out short iStatus)

UserShare.Get_DINT(ushort iUserAddress, out int iStatus)

UserShare.Get_LINT(ushort iUserAddress, out long iStatus)
```

Parameter:

iUserAddress : Address of Variable (1 to 8192)
iStatus : Get the value of Win-GRAF integer variables.

Example:

```
Int64 Dlong_val;
Int16 short_val;
Int32 long_val ;
sbyte sbyte_val;

// Get 64-bit integer value from the Win-GRAF LINT variable with address “7”.
UserShare.Get_LINT(Convert.ToUInt16(7),out Dlong_val);

// Get 32-bit integer value from the Win-GRAF DINT variable with address “8”.
UserShare.Get_DINT(Convert.ToUInt16(8),out long_val);

// Get 16-bit integer value from the Win-GRAF INT variable with address “9”.
UserShare.Get_INT(Convert.ToUInt16(9),out short_val);

// Get 8-bit integer value from the Win-GRAF SINT variable with address “10”.
UserShare.Get_SINT(Convert.ToUInt16(9),out sbyte_val)
```

Demo program:

CD-ROM:

1. R/W analog I/O:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp02
2. R/W internal Boolean, long integer, Timer and Real (floating-point) values:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp03

14.5.3 R/W Real variable Functions

■ **Get_REAL** ■ **Get_LREAL**

Description:
These functions are to get 32-bit REAL and 64-bit double from the Win-GRAF.

Syntax:

UserShare. Get_REAL (System.UInt16 *iUserAddress*, out float *iStatus*)

UserShare. Get_LREAL(ByVal *iUserAddress* As System.UInt16 , out Double *iStatus*)

Parameter:
iUserAddress : Address of Variable (1 to 8192)
iStatus : Get the 32-bit REAL or 64-bit double value.

Example:

```
float float_val;
double double_val;

// Get 64-bit double value from the Win-GRAF LREAL variable with address "7".
UserShare.Get_LREAL(Convert.ToUInt16(7),out double_val);

// Get 32-bit REAL value from the Win-GRAF REAL variable with address "8".
UserShare.Get_REAL(Convert.ToUInt16(8),out float_val);
```

Demo program:
CD-ROM:
1. R/W analog I/O:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp02
2. R/W internal long integer, Timer and Real (floating-point) values :
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_ demo_CSharp03

■ Set_REAL ■ Set_LREAL

Description:

These functions are to set 32-bit REAL and 64-bit double value to the Win-GRAF REAL/LREAL variable.

Syntax:

UserShare. Set_REAL (ushort *iUserAddress* , float *iStatus*)

UserShare. Set_LREAL(ushort *iUserAddress* , Double *iStatus*)

Parameter:

iUserAddress : Address of Variable. (1 to 8192)

iStatus : Set the 32-bit REAL or 64-bit double.

Example:

```
// Set a 64-bit double value "11234.234567" to the Win-GRAF LREAL variable with address "7"
UserShare.Set_LREAL(Convert.ToUInt16(7),Convert.ToDouble(11234.234567));

// Set a 32-bit REAL value "123.12" to the Win-GRAF REAL variable with address "2".
UserShare.Set_REAL(Convert.ToUInt16(8), Convert.ToSingle (123.12));
```

Demo program :

CD-ROM:

1. R/W analog I/O:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp02
2. R/W internal long integer, Timer and Real (floating-point) values :
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp03

14.5.4 R/W String variable Functions

Set_STRING

Description:
This function is to get a Win-GRAF String variable.

Syntax:

UserShare.Set_STRING (ushort addr , Byte [] msg)

Parameter:
addr : Address of Variable (1 to 1024)
msg[] : Get the string value.

Example:

```
String str_val;
Byte[] msg;

// Get the String value of the Win-GRAF String variable with address “7”.
msg= unicode_to_byte_array(str_val);
UserShare.Set_STRING(Convert.ToUInt16(7),msg );

//Convert String to byte array.
private byte[] unicode_to_byte_array(string msg)
    {
      byte[] tmpbuf;
      if (msg.Length > 255)
        return null;

      tmpbuf = Encoding.GetEncoding("UTF-8").GetBytes(msg);
      return tmpbuf;
    }
```

Demo program:
CD-ROM:
1. R/W String variable :
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp04

Get_STRING

Description:

This function is to set a String value to the Win-GRAF String variable.

Syntax:

```
UserShare.Set_STRING (ushort  addr , Byte []  msg)
```

Parameter:

addr : Address of Variable. (1 to 1024)

msg[] : Set the string value.

Example:

```
String str_val= “Hello World”;
Byte[] msg;

// Set a string value “Hello World” to the Win-GRAF String variable with address “7”.
UserShare.Get_STRING(Convert.ToUInt16(7),msg );
str_val= byte_array_to_unicode(msg);

//Convert byte array to String
private string byte_array_to_unicode(byte[] buf)
 {
        string tmpmsg;
        if (buf.Length > 255)
           return null;

        tmpmsg = Encoding.GetEncoding("UTF-8").GetString(buf, 0, buf.Length);
        return tmpmsg;
 }
```

Demo program:

CD-ROM:

1. R/W String variable : \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp04

14.5.5 How to Use C# to Convert Win-GRAF String Variable

Before .NET program write to Win-GRAF String variable. The String-type has to convert to byte array. (According your .NET program Encode. Ex: UTF-8) If you need to read Win-GRAF String variable. Then you have to convert byte array to String. There is an C# example to show how to convert each other.

Example (Encode is UTF-8):

```
//Convert String to byte array
private byte[] unicode_to_byte_array(string msg)
    {
        byte[] tmpbuf;
        if (msg.Length > 255)
            return null;

        tmpbuf = Encoding.GetEncoding("UTF-8").GetBytes(msg);
        return tmpbuf;
    }

//byte array to string
private string byte_array_to_unicode(byte[] buf)
 {
        string tmpmsg;
        if (buf.Length > 255)
            return null;

        tmpmsg = Encoding.GetEncoding("UTF-8").GetString(buf, 0, buf.Length);
        return tmpmsg;
  }
```

Chapter 15 Using eLogger HMI in the Win-GRAF PAC

"eLogger" is an HMI development tool developed by ICP DAS. It features an easy-to-use graphical user interface (GUI), not only supports the Local HMI but also the Web HMI. Users can design their own HMI pages by using eLogger, and then both the eLogger HMI and Win-GRAF softLogic can be executed in the same Win-GRAF PAC.

15.1 The Win-GRAF Project

1. Copy the demo file from the CD-ROM of Win-GRAF PAC to your PC desktop, and then upzip the file. \napdos\Win-GRAF\demo-project\Soft-GRAF-demo**demo_faq018_all.zip**.

Demo project for the Win-GRAF and eLogger:

Win-GRAF Project	eLogger Project	Description
eL01.zip	eL_01.wez	Designing the local and web HMI pages
eL02.zip	eL_02.wez	Configuring a control button that can be instantly ON and then OFF
eL03.zip	eL_03.wez	Reading or modifying the PAC's date and time and doing some control

2. Restore the Win-GRAF project (.zip) to the Win-GRAF workbench on PC (see Section 13.1).
3. Download the Win-GRAF project (e.g., eL01) to the PAC (see Section 2.3.4 and Section 2.3.5).

For more description about the Win-GRAF and eLogger HMI application, visit:
www.icpdas.com > Support > FAQ > Win-GRAF Soft-Logic PAC > FAQ-018 or
http://www.icpdas.com/root/support/faq/win-graf.php

15.2 The eLogger Project

The user can download and setup the eLogger from the eLogger webpage, and copy all eLogger projects (.wez) to **C:\ICPDAS\eLogger\eLogger_Developer\Project** on your PC, and then download the project (e.g., eL_01) to the PAC.

eLogger web page:
www.icpdas.com > Product > Solution > Software > SCADA/HMI > eLogger
http://www.icpdas.com/root/product/solutions/software/scada_hmi/elogger/elogger.html

eLogger software:
http://ftp.icpdas.com/pub/cd/winpac/napdos/elogger/setup/

eLogger manuals:
http://ftp.icpdas.com/pub/cd/winpac/napdos/elogger/document/

Chapter 16 Redundancy

16.1 Features and Architecture

The ICP DAS Win-GRAF PAC - XP-8xx8-CE6 series support redundancy:

One redundant system is composed by two Win-GRAF PACs that one PAC's rotary switch is set to 7 (means Main-PAC) and the other one is set to 9 (means Backup-PAC). When one of them is damaged or crashed or need to release its control-right by user-defined event, the PAC control-right will automatically switch to the healthy one.

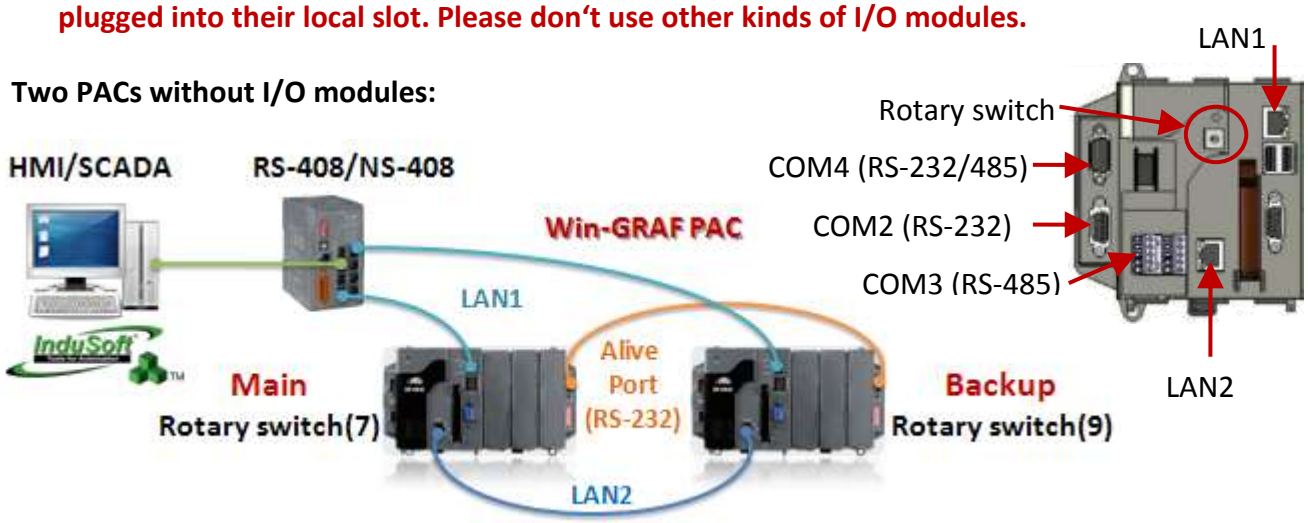
Features of the Win-GRAF redundancy

1. **Better safety:**
   There are three communication cables (LAN1, LAN2 and Alive Port) connected between two PACs. The redundant system will still control the process well, even if one or two cables are broken or disconnected. As long as one of the three communication cables is fine, the Win-GRAF redundant PACs can still work well with the process.
2. **Unique Public IP:**
   The Win-GRAF redundant system provides a unique public IP address for SCADA/HMI to access it without needing to determine which one is the Active IP.
3. **Easy maintenance:**
   If one of the redundant PACs is damaged someday after starting the process, you can remove the damaged one (**Note:** **Do not** shutdown or dismounting the other healthy PAC, keep it running). And then take another spare Win-GRAF PAC with the same model (or a repaired PAC) without downloading the Win-GRAF application, simply adjust its rotary switch to a proper position and then connect all required communication cables (e.g., LAN1, LAN2, Alive port and I/O). Make sure that the original, healthy PAC is still working properly and then power up the spare PAC. Then, the healthy PAC will automatically copy the Win-GRAF app and all its redundant data to that new PAC which is just online. It is easier for maintenance and installation, the operator don't have to worry about whether to install the Win-GRAF app because the healthy PAC will automatically do it for the new online PAC.
   Exception: Except the Win-GRAF app if there are a few apps, such as the C, VB.net, C# app or eLogger HMI app, running in the redundant system, these apps need to pre-installed to the spare Win-GRAF PAC (or a repaired PAC) before installing this PAC to the redundant system.
4. **Easy to design the application:**
   The user has to do is design the application program. Not necessary to specify what data should be redundant between two PACs. The Win-GRAF redundant system will automatically deliver them to the other PAC.
5. **Users can design some safety in the app:**
   For example, if the Active PAC's LAN1 is disconnected (causes the SCADA unable to connect to) or a RS-485 Port is disconnected or damaged, and so on. The user's app can test these events and then switch control right to the other healthy PAC.
6. **I/O Redundancy:**
   If the user chooses iDCS-8830 series I/O, both the PAC and I/O modules can support redundancy.

<u>The architecture of the Win-GRAF redundant system</u> (using XP-8xx8-CE6 as an example):

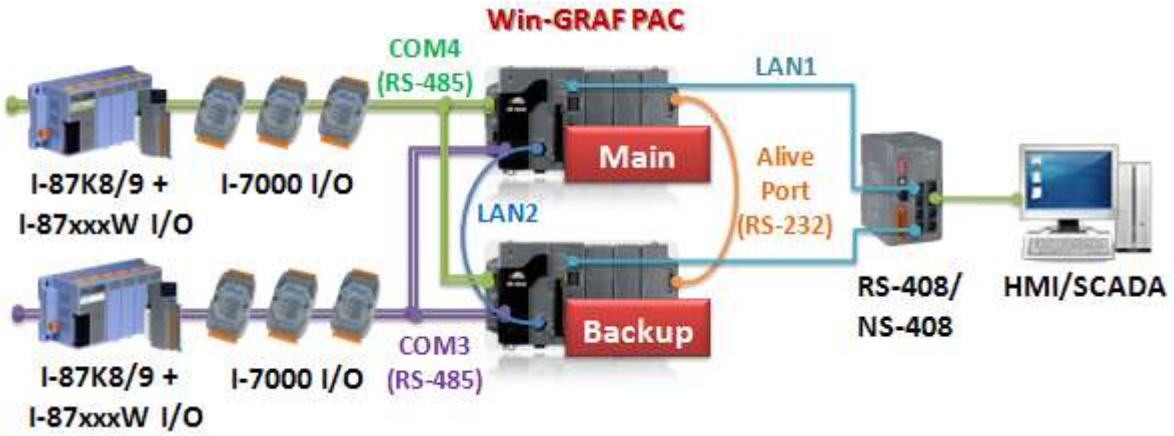
<u>Note:</u> The Win-GRAF redundant PACs support RS-485/RS-422 expansion boards (I-8142i/ I-8144i) plugged into their local slot. Please don't use other kinds of I/O modules.

1. Two PACs without I/O modules:



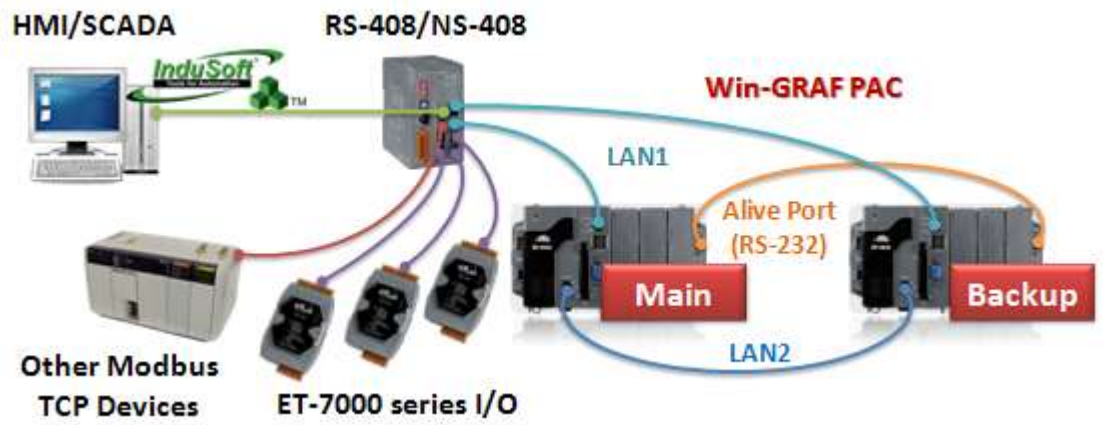
<u>Note:</u> <u>LAN1</u>: Normal Ethernet Cable, <u>LAN2</u>: Ethernet Crossover Cable, <u>Alive Port</u>: RS-232 Crossover Cable.

2. Two PACs are equipped with DCON I/O modules:



<u>Note:</u> <u>LAN1</u>: Normal Ethernet Cable, <u>LAN2</u>: Ethernet Crossover Cable, <u>Alive Port</u>: RS-232 Crossover Cable.
<u>COM3, COM4 (RS-485)</u>: Data+ to Data+ ; Data- to Data- .

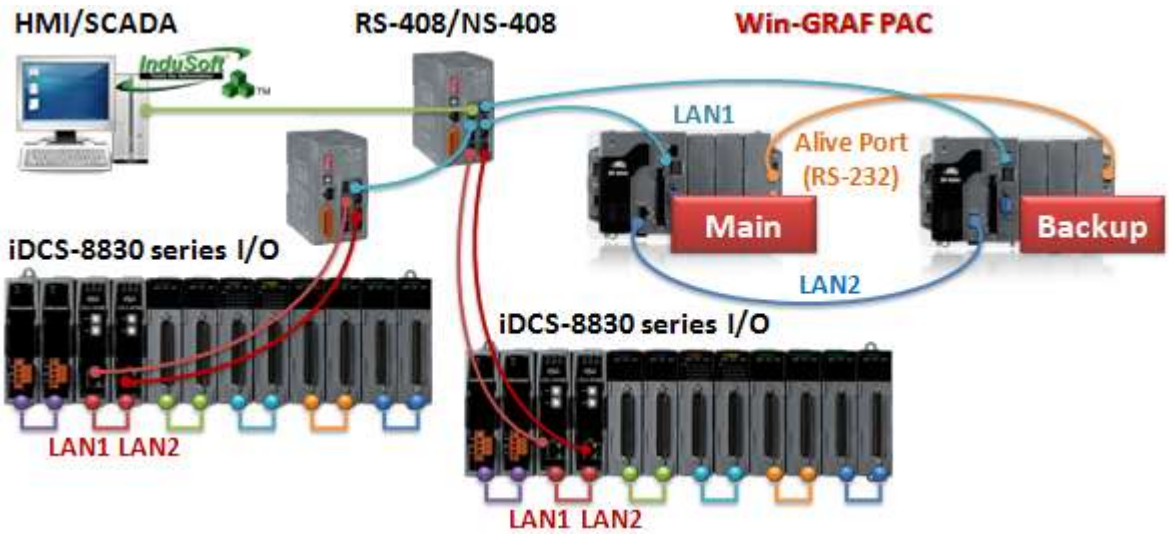
3. Two PACs are equipped with Modbus TCP I/O modules:



<u>Note:</u> <u>LAN1</u>: Normal Ethernet Cable, <u>LAN2</u>: Ethernet Crossover Cable, <u>Alive Port</u>: RS-232 Crossover Cable.

4. Two PACs are equipped with iDCS-8830 I/O modules:

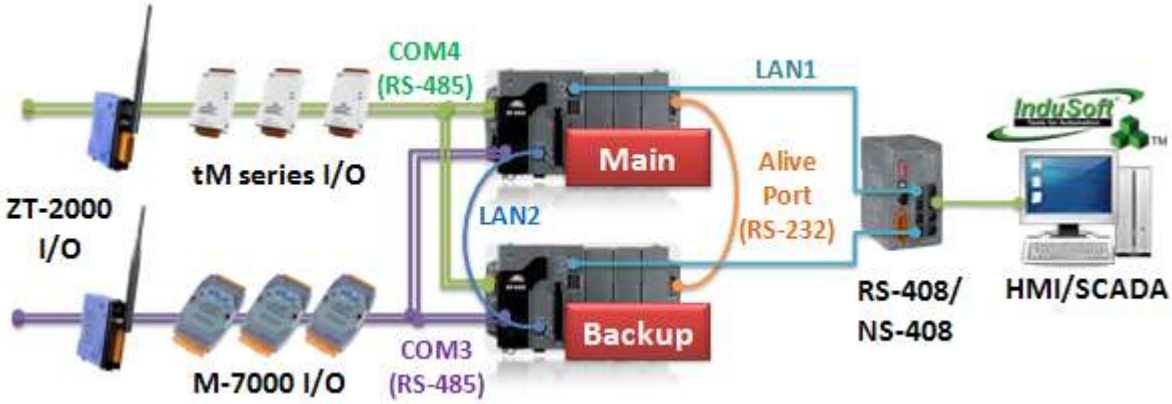
This type of achievement provides both CPU and I/O module redundancy.



Note: LAN1 (PAC), LAN1 (iDC-8830), LAN2 (iDC-8830): Normal Ethernet Cable.
LAN2 (PAC): Ethernet Crossover Cable, Alive Port: RS-232 Crossover Cable.

Note: Each pair of redundant I/O modules that plugged into the iDCS-8830 must have the same model numbers.

5. Two PACs are equipped with other Modbus RTU/ASCII I/O modules:



Note: LAN1: Normal Ethernet Cable, LAN2: Ethernet Crossover Cable, Alive Port: RS-232 Crossover Cable.
COM3, COM4 (RS-485): Data+ to Data+ ; Data- to Data- .

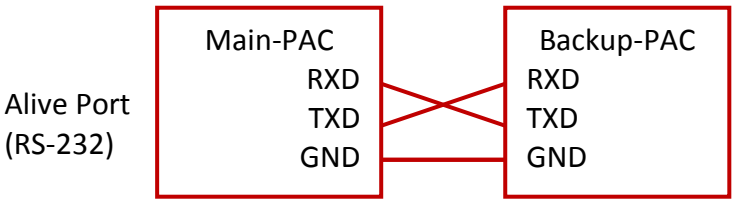
6. It can also equip with two (or more) kinds of I/O modules such as item (2) to (5).

The Win-GRAF redundant system is composed by two PACs. Users need to set one PAC's rotary switch to 7 (called Main-PAC) and set the other one to 9 (called Backup-PAC). **Do not** use two Main PACs or two Backup PACs to make up a redundant system.

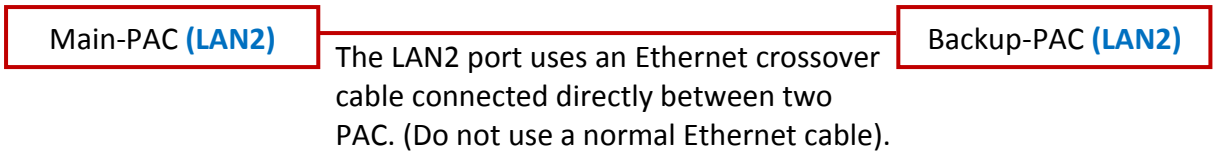
16.2 Important Communication Ports and Installation Notes

The Win-GRAF redundant PACs require the following three communication ports to communicate with each other.

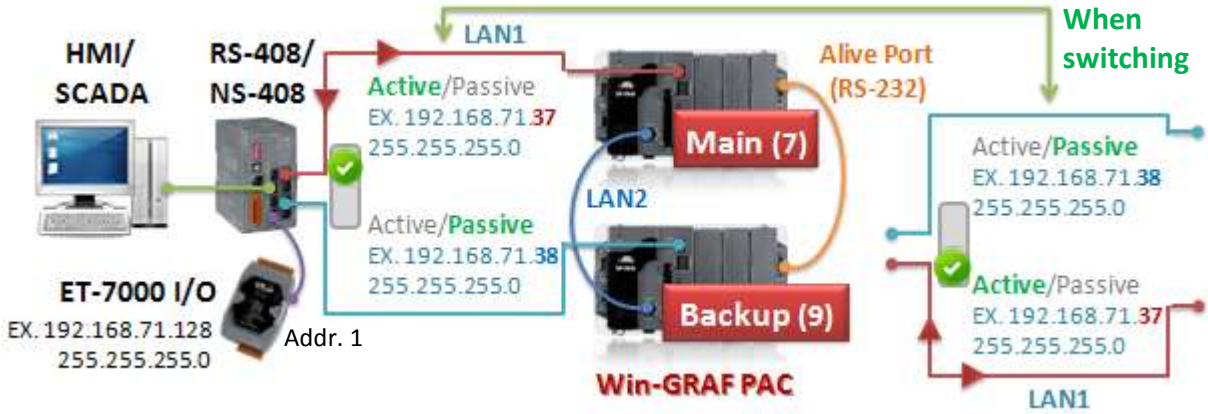
1. **Alive Port:**
   The Win-GRAF redundant PACs use one RS-232 Port as the Alive Port (also called Heart-beat Port). This Alive Port must use a RS-232 crossover cable (or NULL Modem Cable), which link with each other as the following figure.



2. **Replication Port:**
   The Win-GRAF redundant PACs use their Ethernet Port (LAN2) as a Replication Port. Both of PAC's LAN2 ports use an Ethernet crossover cable to transfer redundant data. Do not use any Ethernet Switch or Hub between them, otherwise it may cause an error or timeout. The LAN2 ports are based on fast Ethernet and dedicated Ethernet ports in order to avoid collisions. So don‘t connect any external devices, Switches, and Hubs to these two PAC’s LAN2.



3. **Public IP Port:**
   The Win-GRAF redundant PAC’s Ethernet Port (LAN1) must connect an Ethernet Switch via a normal Ethernet cable. After that, it can be used to communicate with SCADA/HMI or connect and control external Modbus TCP I/O modules, devices or other Ethernet devices. The LAN1 port can switch its IP address automatically. If the PAC is Active, the LAN1 IP address will switch to the "Active_IP" address which defined in the user’s Win-GRAF project. And if the PAC is Passive, the LAN1 IP address will switch to the "Active_IP+1" address automatically. The SCADA/HMI can use the "Active_IP" address to communicate with the Win-GRAF redundant system.



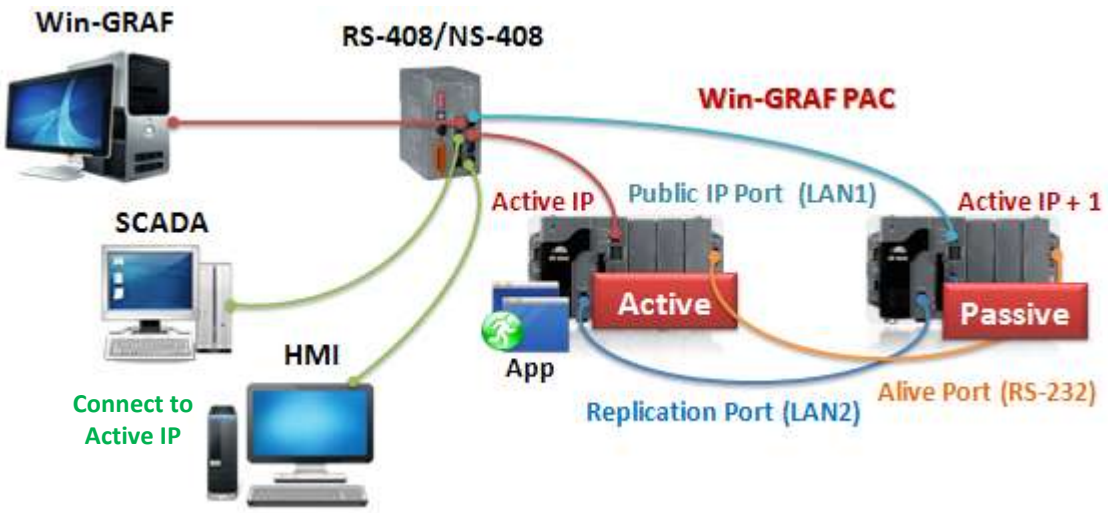
PAC Installation Notes (Very Important):

1. Before power up PACs, make sure one PAC's rotary switch is set to 7 and the other one is set to 9. The redundant system will be crazy due to the wrong settings.

2. When installing the Win-GRAF redundant system at the application field, make sure the following three cables are connected properly (connect all required cables, such as RS-485) before power up PACs. If user power up PACs before connecting these three communication cables, the redundant system will be out of control.

   A. Connect both of the PAC's Alive ports by using a crossover cable.
   B. Connect both of the PAC's LAN2 ports by using an Ethernet crossover cable.
      (Do not use any Ethernet Switch/Hub between LAN2 ports.)
   C. Connect both of the PAC's LAN1 ports to an Ethernet switch by using a normal Ethernet cable.

3. If only one healthy PAC of the redundant system is working properly at the application field, do not power-off or shut it down. Before user power up the other PAC that will be installed into the system, follow the step1 and step2 as mentioned above to set up it first.



Only the Active PAC (i.e., PAC got the control-right) can run the Win-GRAF application. The Passive PAC will not run the Win-GRAF application. It simply receives the redundant data from the Active PAC and wait for getting control-right in the future.

16.3 Description of Win-GRAF Demo Projects

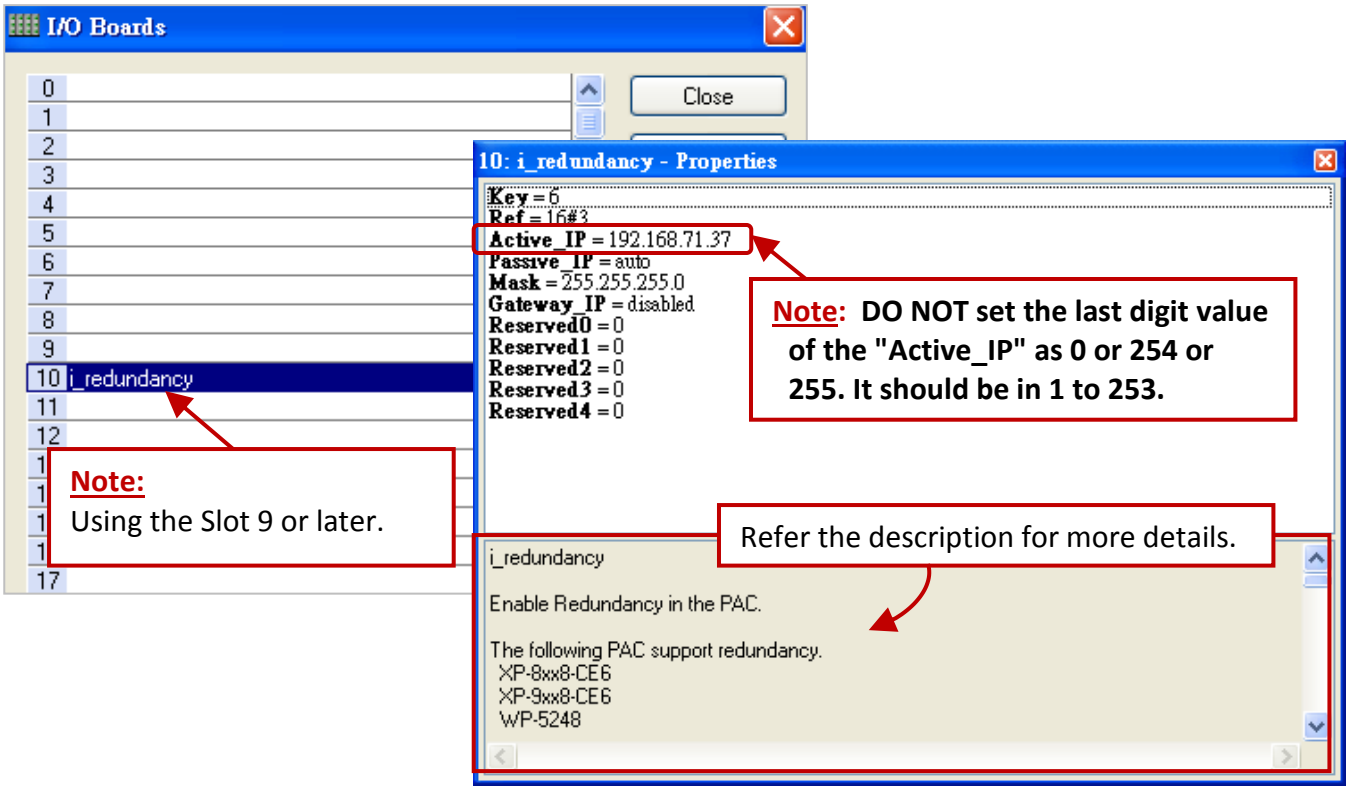
The shipping CD of the Win-GRAF PAC provides these three demo projects – "demo_RDN_1.zip", "demo_RDN_2.zip" and "demo_RDN_3.zip" – related to the redundant system. Refer Chapter 12 to restore these files into the Win-GRAF Workbench.

Project Name	Description
demo_RDN_1	Two XP-8xx8-CE6 PACs, using their COM3 to connect three DCON I/O modules.
demo_RDN_2	Two XP-8xx8-CE6 PACs without connecting any I/O module.
demo_RDN_3	Two XP-8xx8-CE6 PACs, using their LAN1 to connect a ET-7050 (Modbus TCP I/O module) through one Ethernet switch.
demo_RDN_4	Two XP-8xx8-CE6 PACs, using their LAN1 to connect an iDCS-8830 (Both PAC and I/O are redundant) through one Ethernet switch.

The following sections will describe the **"demo_RDN_2"** program.

16.3.1 "I/O Board" Settings

demo_RDN_2, demo_RDN_3, demo_RDN_4::
To use redundancy on the PAC, first link the "i_redundancy" in the "I/O Board" window.
(Refer Chapter 4).



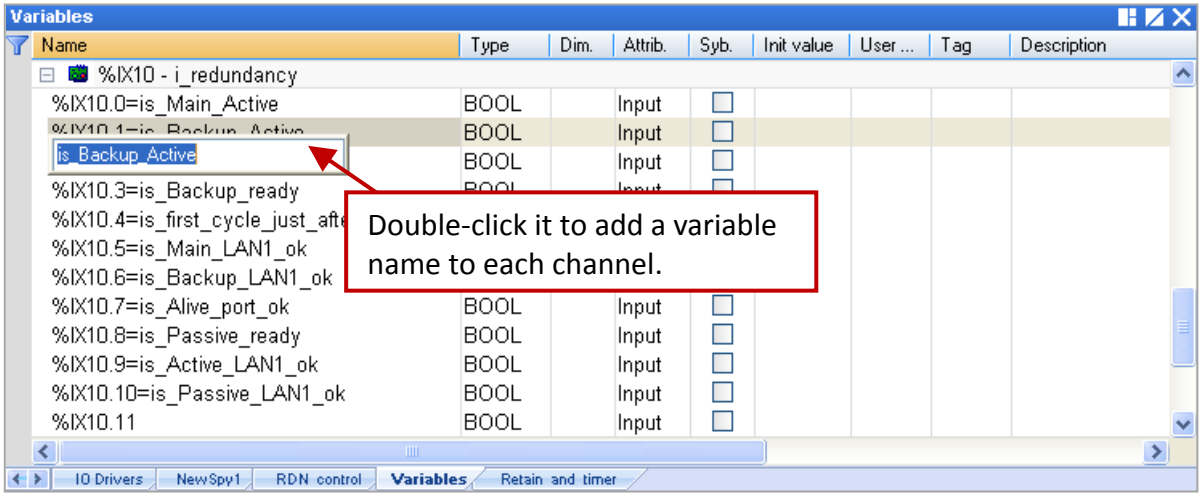
Parameters:

Active_IP: The redundant system provides a public IP address for some HMI/SCADA to communicate. (**Note: DO NOT** set the last digit value of the "Active_IP" as 0 or 254 or 255. It should be in 1 to 253.)

Passive_IP: Auto, means the LAN1 IP address of the current Passive PAC, it will be automatically set as Active_IP +1 (e.g., if the "Active_IP" is set as "192.168.71.37", the "Passive_IP" will automatically set as "192.168.71.38“)

Mask: The most common settings are either 255.255.255.0 or 255.255.0.0 (depends on the network environment).

After linking the “i_redundancy” in the “I/O Boards” window, it will auto add 12 "BOOL” input channels in the “Variables” window that can be used to display the current state of the redundant system.



Ch.0 (is_Main_Active): Is the Main-PAC (rotary switch: 7) active now?
TRUE: Active , FALSE: Passive

Ch.1 (is_Backup_Active): Is the Backup-PAC (rotary switch: 9) active now?
TRUE: Active , FALSE: Passive

Ch.2 (is_Main_ready): Is the Main-PAC ready?
If Ch.2 returns FALSE. The possible reason could be the following.
(1) The Ethernet cable (LAN2) between Main and Backup PAC is broken.
(2) The Main PAC is dead or damaged.
(3) The rotary switch of the Main PAC is not set at 7.

Ch.3 (is_Backup_ready): Is the Backup-PAC ready?
If Ch.3 returns FALSE. The possible reason could be the following.
(1) The Ethernet cable (LAN2) between Main and Backup PAC is broken.
(2) The Main PAC is dead or damaged.
(3) The rotary switch of the Main PAC is not set at 9.

Ch.4 (is_first_cycle_just_after_switch): For Active PAC only.
True: Now is in the first cycle just after switching.
False: Now is not in the first cycle after switching.

Ch.5 (is_Main_LAN1_ok): Is the LAN1 port of the Main-PAC ok?
TRUE: OK , FALSE: Fail or Ethernet cable is disconnected.

Ch.6 (is_Backup_LAN1_ok): Is the LAN1 port of the Backup-PAC ok?
TRUE: OK , FALSE: Fail or Ethernet cable is disconnected.

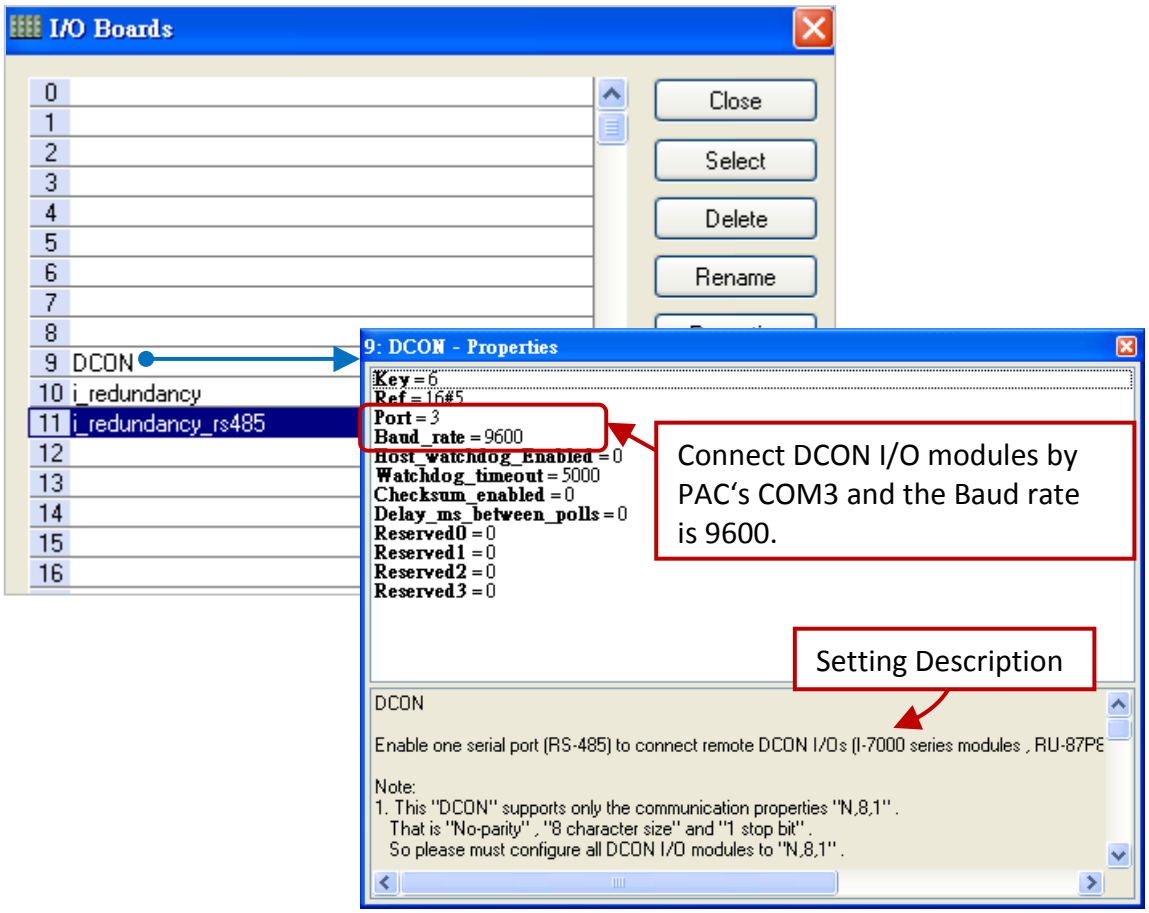
Ch.7 (is_Alive_port_ok): True : The communication of the Alive Port is ok.
False: The communication of the Alive port fails or the Passive PAC is dead or damaged.

Ch.8 (is_Passive_ready): Is the Passive PAC ready now?
If Ch.8 returns FALSE. The possible reason could be the following.
(1) The Ethernet cable (LAN2) between Main and Backup PAC is broken.
(2) The Passive PAC is dead or damaged.
(3) The rotary switch setting of the Passive PAC is incorrect.

Ch.9 (is_Active_LAN1_ok): Is the LAN1 port of the Active-PAC ok?
TRUE: OK , FALSE: Fail or Ethernet cable is disconnected.

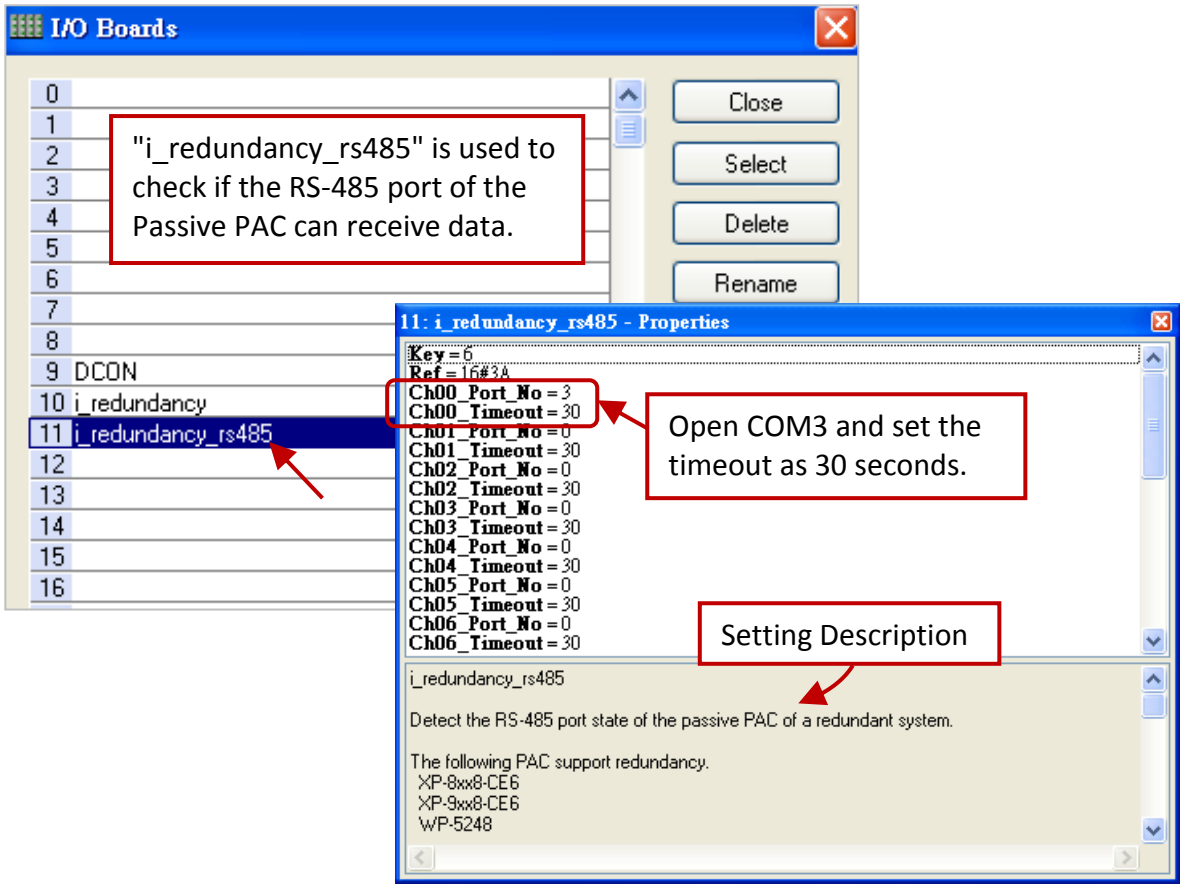
Ch.10 (is_Passive_LAN1_ok): Is the LAN1 port of the Passive-PAC ok?
TRUE: OK , FALSE: Fail or Ethernet cable is disconnected.

demo_RDN_1: To connect DCON I/O modules via PAC's COM3 (RS-485).



Important Notice:

1. Please must also use the "i_redundancy" or the "i_redundancy_rs485" will not work.
2. The "i_redundancy_rs485" will only open the related RS-485 ports to receive data in the passive PAC. It doesn't send any data.
3. The "i_redundancy_rs485" is used to detect whether the Passive PAC's RS-485 port can receive data.



Parameters:

Ch00_Port_No ~ Ch15_Port_No :

The used RS-485 port number of the Passive PAC. Can be 0 or 1 to 33 depends on the PAC model. Set 0 means disable it.

Ch00_Timeout ~ Ch15_Timeout :

The unit is second. Can be 1 to 60 seconds.

If there is no data received in the timeout interval of the related RS-495 port, the status will reset as FALSE.

16-ch Boolean Inputs :

It used to represent state of RS-485 ports in the passive PAC.

TRUE : The related RS-485 port open ok and can receive data.

FALSE: The related RS-485 port open fail or receive no data in the timeout interval.

16.3.2 Declaring Variables (demo_RDN_2)

Users can view or add variables in the "Variable" window (refer Section 2.3).

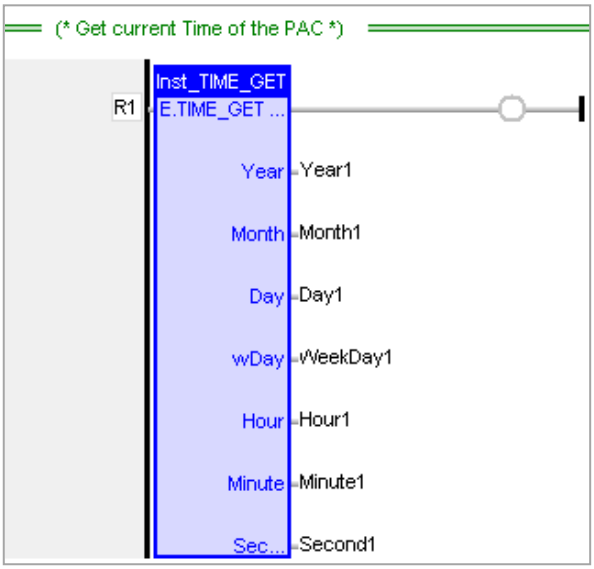
Name	Data Type	Description
Year1	DINT	Used in the “PAC_Time” program: They are used to get the PAC’s system time.
Month1	DINT	
Day1	DINT	
WeekDay1	DINT	
Hour1	DINT	
Minute1	DINT	
Second1	DINT	
Set_new_time	BOOL	Set it as "TRUE" to set up new system time.
Year_to_set	DINT	Used in the “PAC_Time” program: They are used to set the PAC’s system time.
Month_to_set	DINT	
Day_to_set	DINT	
Hour_to_set	DINT	
Minute_to_set	DINT	
Second_to_set	DINT	
DINT_1	DINT	Used in the "Retain_and_timer" program: Set them as retain variables.
DINT_2	DINT	
REAL_1	REAL	
REAL_2	REAL	
TMR_1	TIME	Timer
TMR_2	TIME	
retain_done	BOOL	TRUE: Retain variables are well set up; FALSE: Not set up yet.
on_line_change_cycle	DINT	Non-zero, means this is the first cycle just after On-Line change.
tmp_bool	BOOL	It used to return the Retain status.
TMR_1_last_state	BOOL	TRUE: Ticking ; FALSE: Sleeping.
TMR_2_last_state	BOOL	TRUE: Ticking ; FALSE: Sleeping.
To_tick_TMR_1	BOOL	Set it as TRUE to start TIMER1.
To_tick_TMR_2	BOOL	Set it as TRUE to start TIMER2.
To_stop_TMR_1	BOOL	Set it as TRUE to stop TIMER1.
To_stop_TMR_2	BOOL	Set it as TRUE to stop TIMER2.

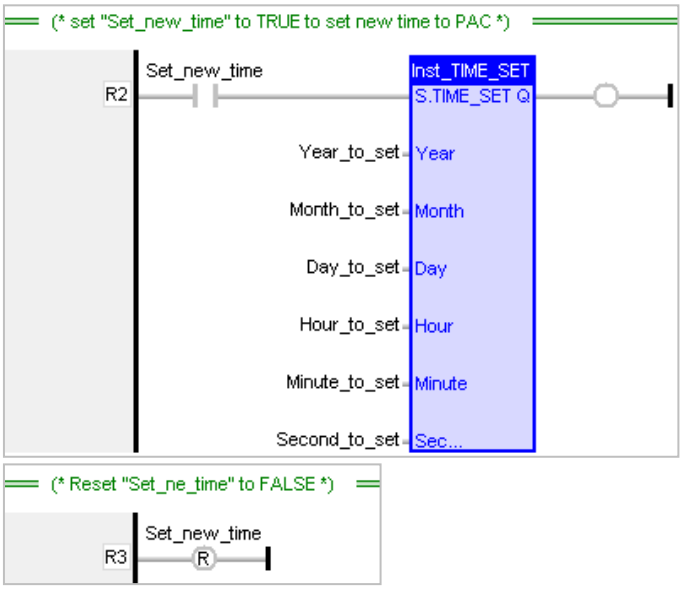
16.3.3 Introduction of the "demo_RDN_2" Project

This project includes one LD program and one ST program.

LD Program – “PAC_Time”

It used to get/set the system time of PAC.





LD Program – “RDN_control”

When an error occurs on the Active PAC’s LAN1 and if the Passive PAC is ready and its LAN1 is healthy, then the Active PAC will wait for a short time to reboot, and then the other PAC will take the control-right.

ST Program – "Retain_and_timer"

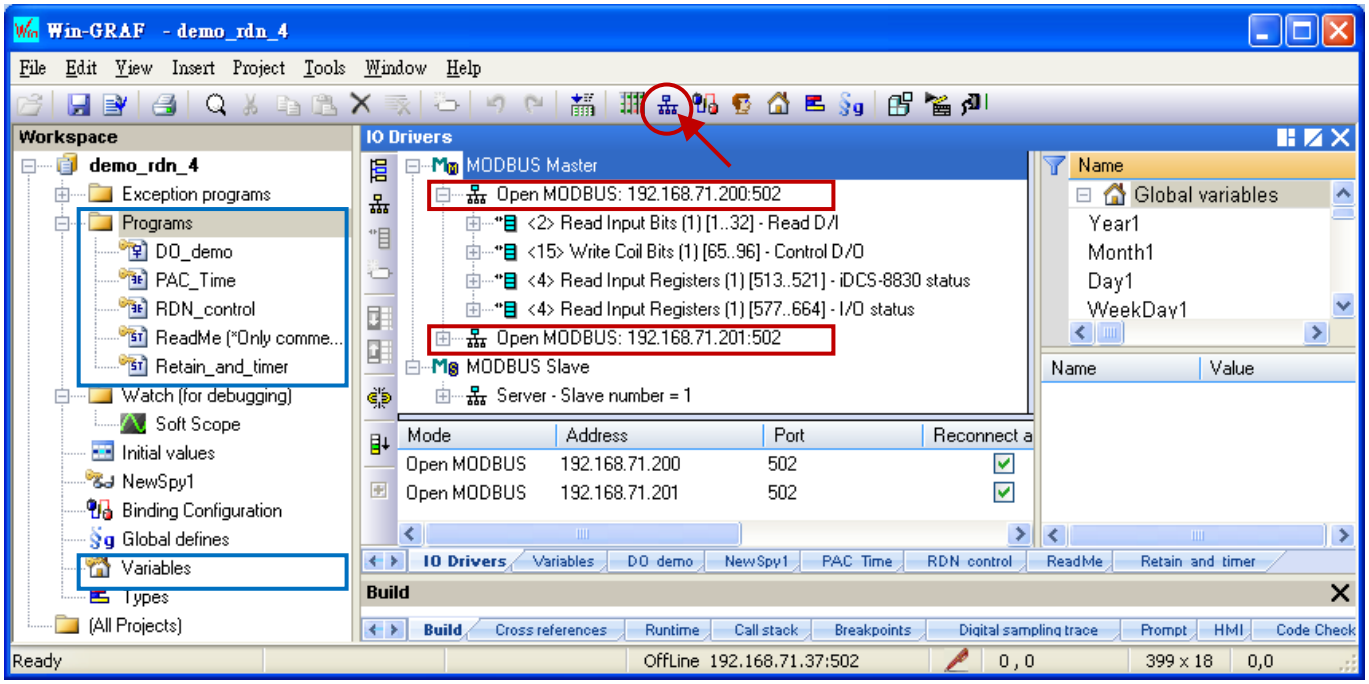
```
(*   "on_line_change_cycle" is declared as DINT (nonezero means  it is in the cycle jsut
     after doing on line change) .
     "retain_done" is declared as BOOL and inited as FALSE .
     "tmp_bool" is declared as BOOL.  *)

on_line_change_cycle := GetSysInfo (_SYSINFO_CHANGE_CYCLE) ;
if (retain_done = FALSE) or
  (is_first_cycle_just_after_switch = TRUE) or
  (on_line_change_cycle <> 0) then
    retain_done := TRUE ;  (*just do it one time *)
    tmp_bool := Retain_Var( DINT_1 , 1) ; (* retain a DINT variable *)
    tmp_bool := Retain_Var( DINT_2 , 2) ;
    tmp_bool := Retain_Var( REAL_1 , 3) ; (* retain a REAL variable *)
    tmp_bool := Retain_Var( REAL_2 , 4) ;
```

```
        (* if Retain variables havn't been inited yet, use default value *)
    if  (DINT_1 < -1000000) or (DINT_1 > 1000000) or
        (DINT_2 < -2000000) or (DINT_2 > 2000000) or
         (REAL_1 < -9.9E10) or (REAL_1 > 9.9E10) or
        (REAL_2 < -9.9E10) or (REAL_2 > 9.9E10)  then
       DINT_1 := 0 ;
       DINT_2 := 0 ;
       REAL_1 := 0.0 ;
       REAL_2 := 0.0 ;
    end_if ;
end_if ;

(*  is_first_cycle_just_after_switch :
     TRUE  : just in the cycle after switching.
     FALSE : other cycle   *)
if  is_first_cycle_just_after_switch  then

  (*  The Timer ticking state is not auto-redundant. So we have to process them here.
      Ticking timer in the cycle just after switching if its last state is "ticking"   *)
  if  TMR_1_last_state  then
    tStart(TMR_1) ;
  end_if ;
  if  TMR_2_last_state  then
    tStart(TMR_2) ;
  end_if ;
end_if ;

(* Timer operation *)
if  To_tick_TMR_1  then
  To_tick_TMR_1 := FALSE ;
  tStart(TMR_1) ;
  TMR_1_last_state := TRUE ;
end_if ;
if  To_tick_TMR_2  then
  To_tick_TMR_2 := FALSE ;
  tStart(TMR_2) ;
  TMR_2_last_state := TRUE ;
end_if ;
if  To_stop_TMR_1  then
  To_stop_TMR_1 := FALSE ;
  tStop(TMR_1) ;
  TMR_1_last_state := FALSE ;
end_if ;
if  To_stop_TMR_2  then
  To_stop_TMR_2 := FALSE ;
  tStop(TMR_2) ;
  TMR_2_last_state := FALSE ;
end_if ;
```

16.3.4 Introduction of the "demo_RDN_4" Project

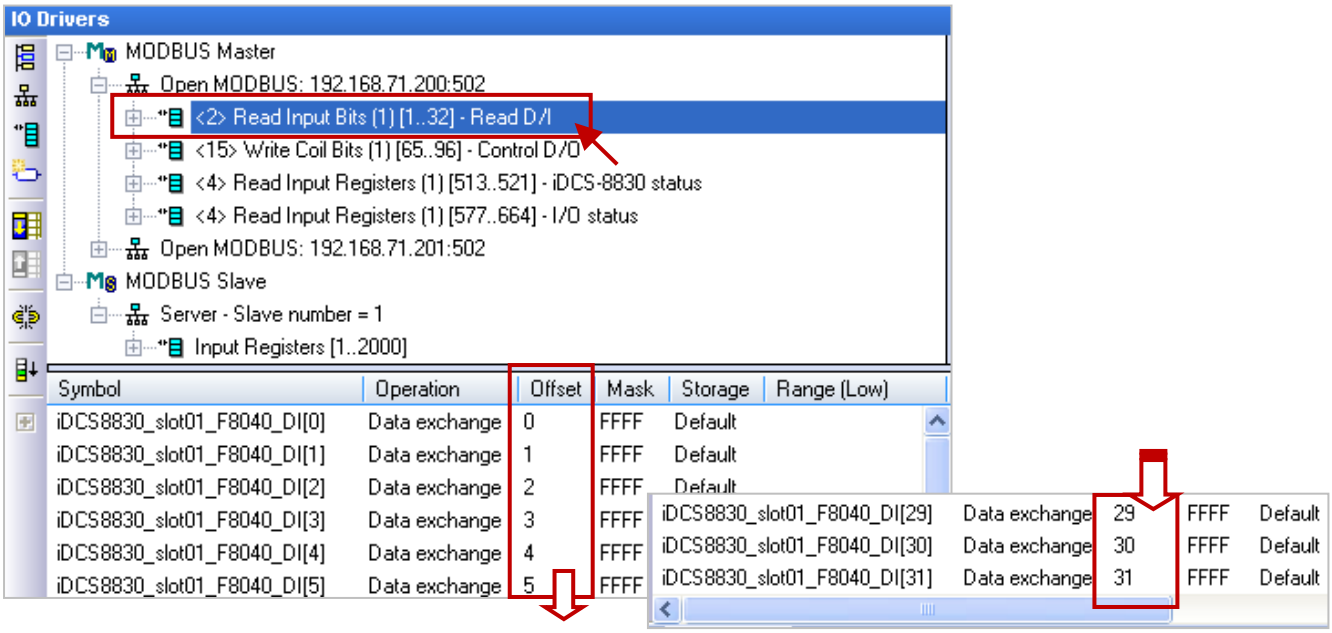
In the "demo_rdn_4" project, you can click the program name to view its content, click "Variables" to see all used variables, or refer Section 16.3.1 to view the I/O Board settings ("i_redundancy").
This section will introduce you the "Modbus Master" function (refer Chapter 5 for more details about operations and the way to set up continuous Offset value for multiple variables).

In this example, we use one redundant I/O expansion unit (iDCS-8830), two redundant DI module (F-8040) plugged in its I/O Solt0, 1, and two redundant DO module (F-8041) plugged in its I/O Solt2, 3. Before starting the test, refer Section 16.4.2 to configure the iDCS-8830 to work properly.
Click the “Open Fieldbus Configuration” button to open the “I/O Drivers” setting window.

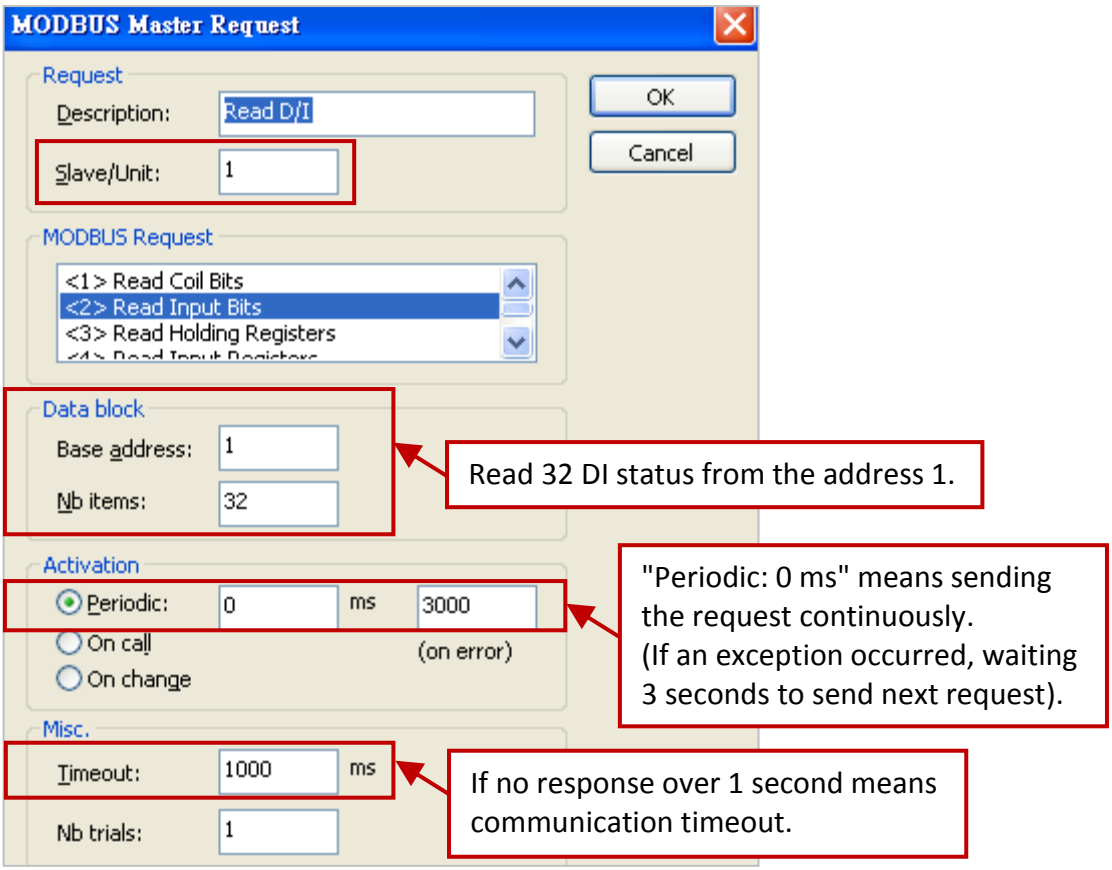


We enable the Modbus Master function to connect two Modbus TCP Slave devices, i.e., each iDCS-8830 redundant I/O unit has two IP addresses, they are "192.168.71.**200**" and "192.168.71.**201**" (Port: 502), which used to read/write the I/O data and status.

<u>Read Digital Inputs (Using two redundant DI module - F-8040 in the slot0, 1)</u>

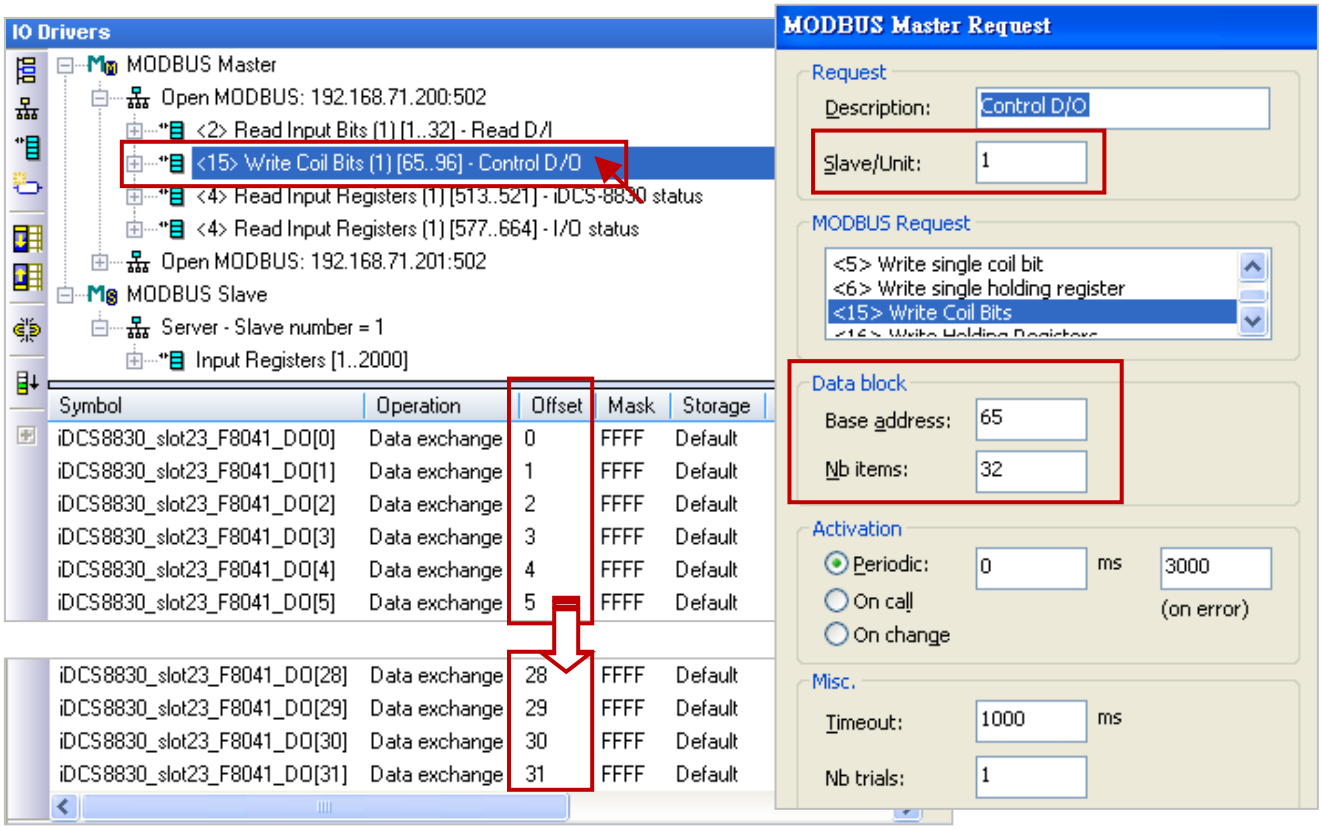


Note: We use an iDCS-8830 redundant I/O unit in this example, refer the iDCS-8000 user manual (CH4 Modbus Addresses Mapping) to know how to input a proper "Base address".
http://ftp.icpdas.com/pub/cd/idcs-8000/usersmanual/fcm-mtcp_software_usermanual_en.pdf



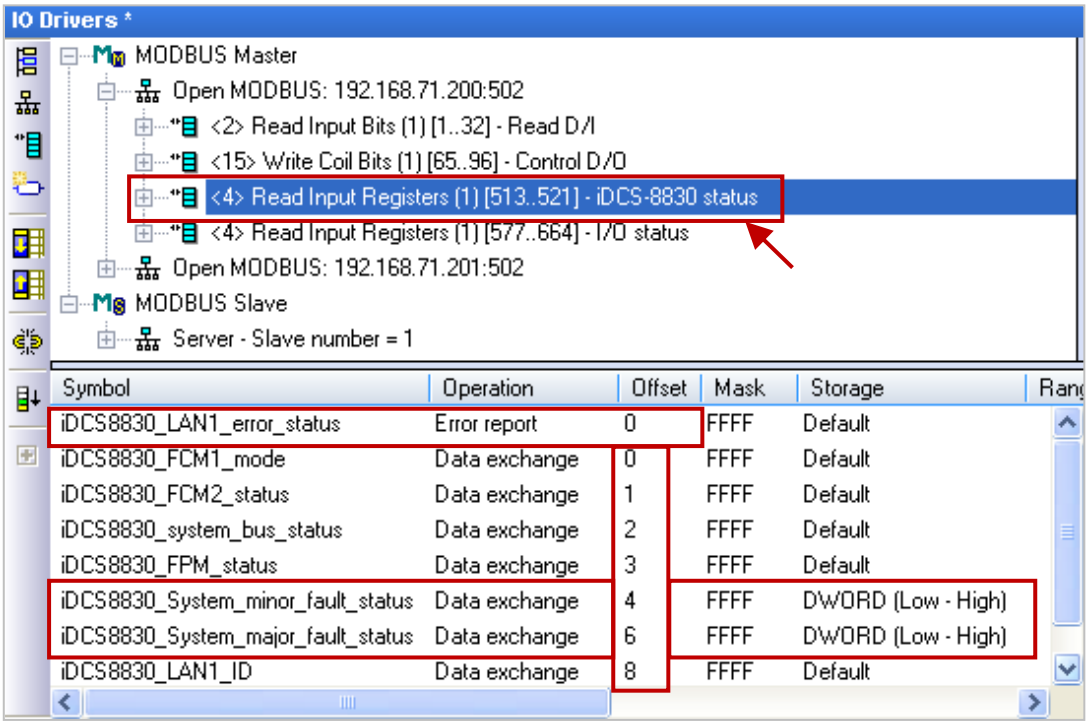
Write to Digital Outputs (Using two redundant DO module - F-8041 in the slot2, 3)

In this example, to write 32 DO status from the address "65" (other settings like the figure above).

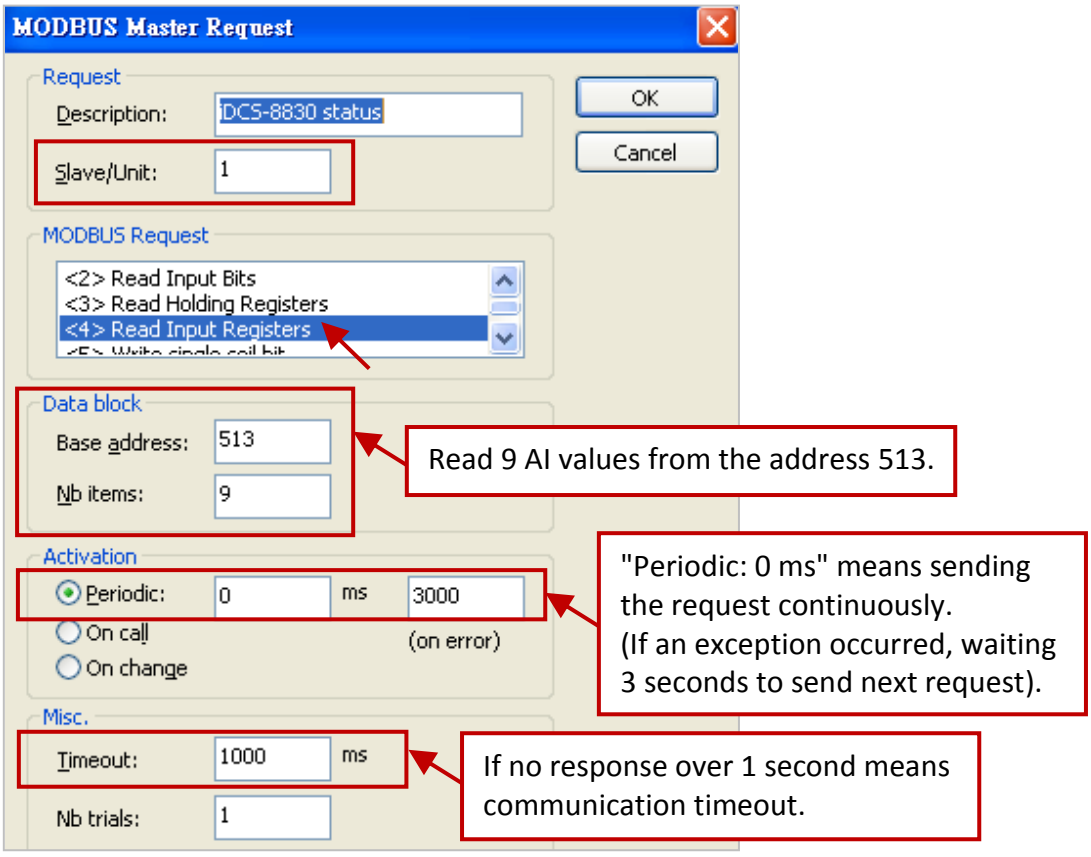


Read the status of the iDCS-8830

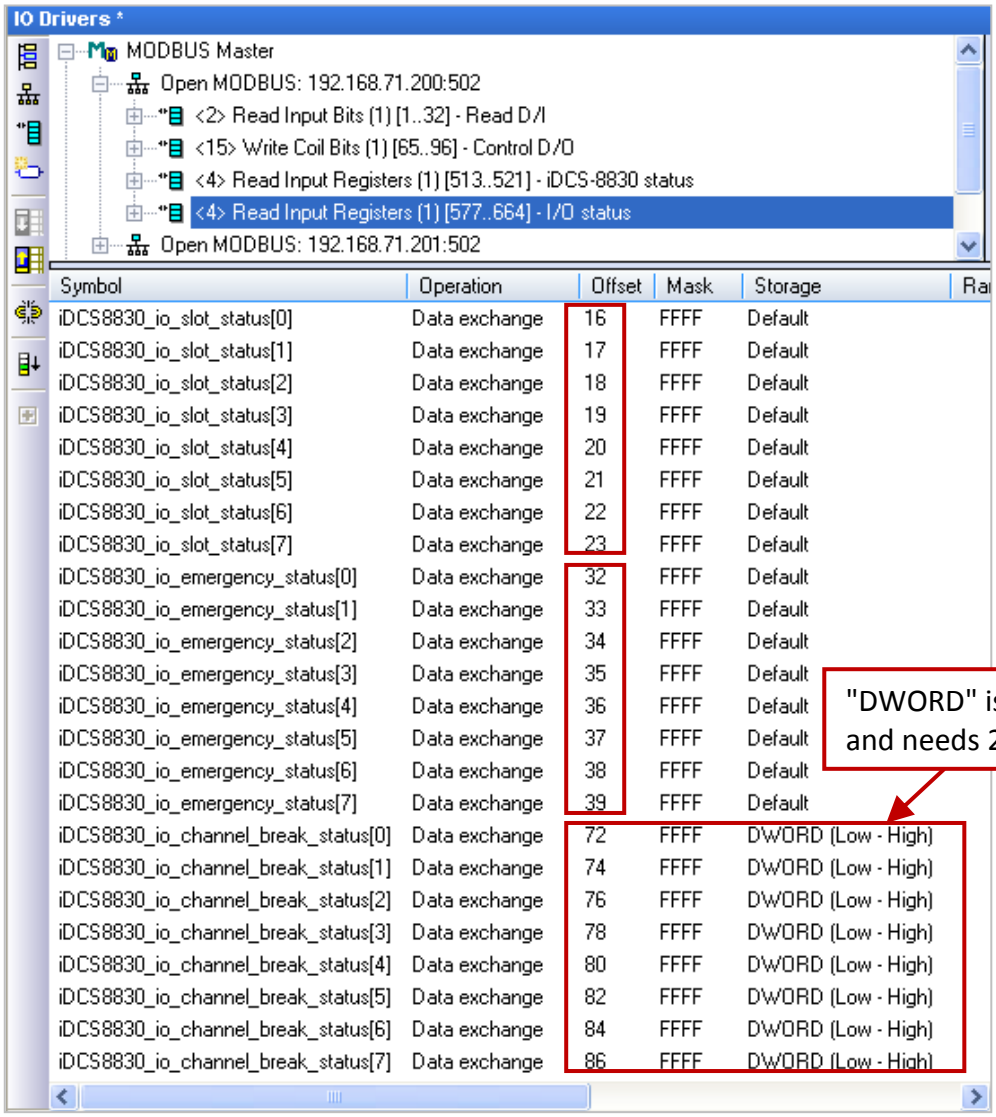
- The “Operation” field of the "iDCS8830_LAN1_error_status" variable is set as “Error report” in order to show an error code when reading failed. The error code will be reset to “0” when reading success. Moreover, its “Offset” field must set as “0”.
- Due to the data type of "iDCS8830_System_minor_fault_status" and "iDCS8830_System_major_fault _status" is "DWORD" (32 bit), the "Offset" must use two Modbus address and the "Storage" must set as "DWORD (Low-High)".



Note: Refer the iDCS-8000 user manual (CH4 Modbus Addresses Mapping) to fill in the "Base address".
http://ftp.icpdas.com/pub/cd/idcs-8000/usersmanual/fcm-mtcp_software_usermanual_en.pdf



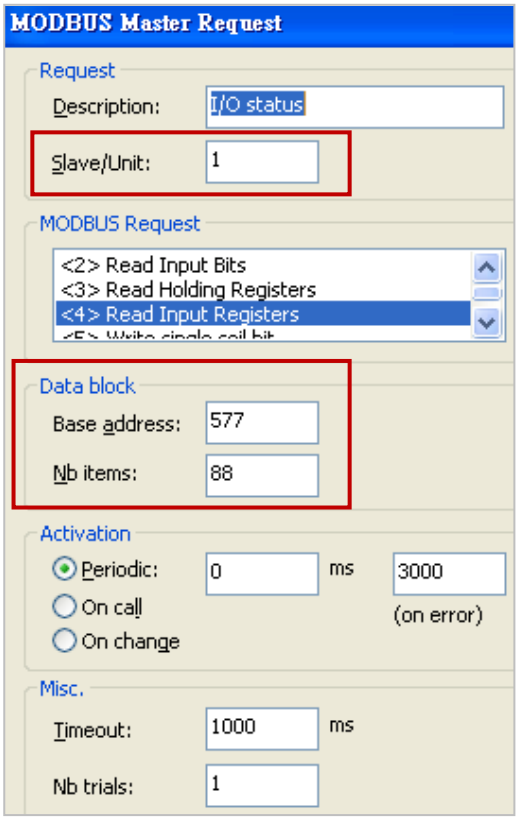
Read the I/O status of the iDCS-8830



Note:

Refer the iDCS-8000 user manual (CH4 Modbus Addresses Mapping) to fill in the "Base address".
http://ftp.icpdas.com/pub/cd/idcs-8000/usersmanual/fcm-mtcp_software_usermanual_en.pdf
In this example, to read 88 AI values from the address 577.

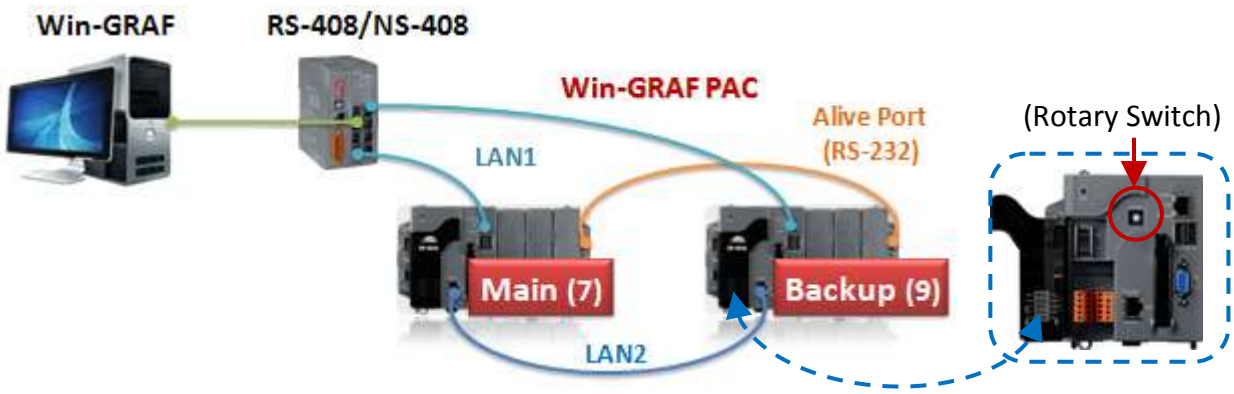
Refer Chapter 5 for more details on Modbus Master settings and refer Chapter 3 for Modbus Slave settings.



16.4 Test Demo Projects

16.4.1 Test the "demo_RDN_2" and "demo_RDN_3" Project

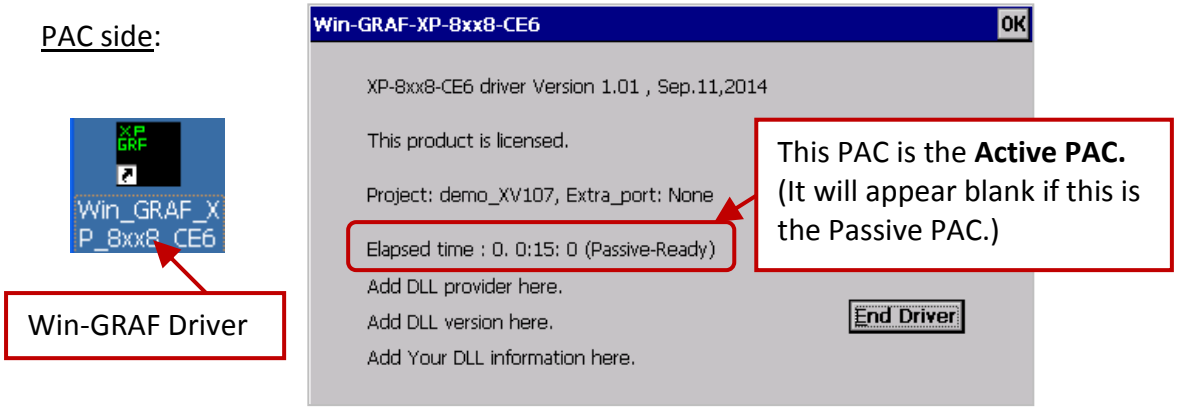
demo_RDN_2: Two PAC (XP-8xx8-CE6) without connecting any I/O modules.



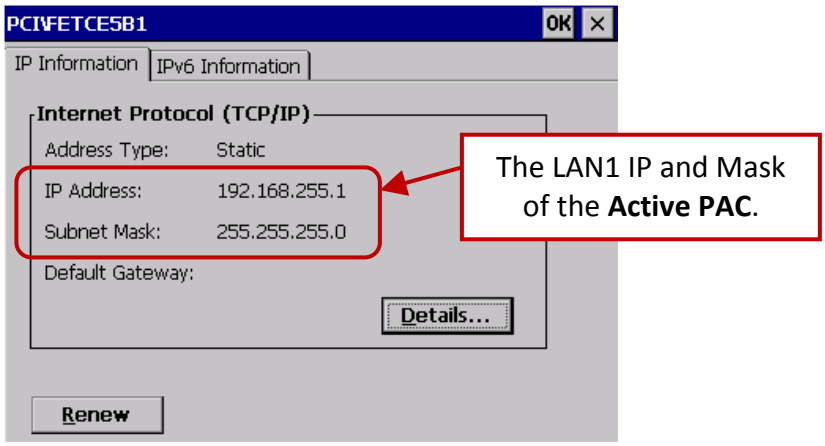
1. Hardware installation (using XP-8xx8-CE6 as an example):
   Refer Section 16.2 – PAC Installation notes, make sure three communication ports of the PAC have been connected properly and the rotary switch is set to 7 (Main-PAC) or 9 (Backup-PAC).

2. If there is no redundancy app in the redundant system yet (that is, no control-right switching procedures), start the Main-PAC (7) first and then start the Backup-PAC (9), in that case the Active PAC will be the Main-PAC.
   (Later, users can run the Win-GRAF driver on a PAC's desktop to see which one is the Active PAC.)

PAC side:



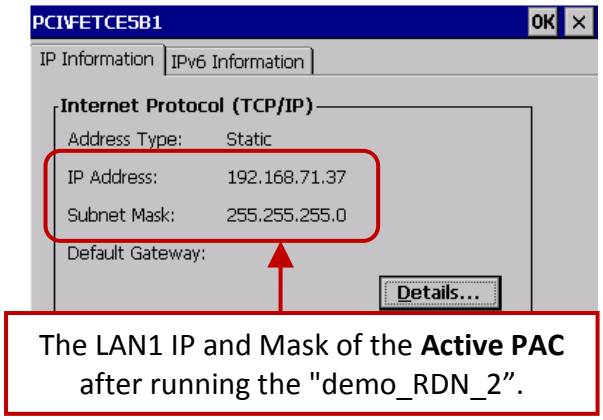
(1) First, look up the LAN1 IP address of the current Active PAC (factory defaults: IP=192.168.255.1, Mask=255.255.255.0, refer Section 1.3), and make sure that your PC is on the same network segment (e.g., IP=192.168.255.x).

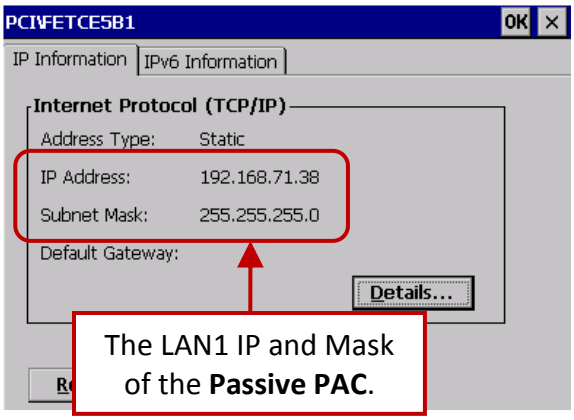


(2) At the first time to download the Win-GRAF redundancy application, users **Must** modify the communication IP address (refer Section 2.3.5 - “Communication Parameters") to the LAN1 IP address of the current Active PAC.

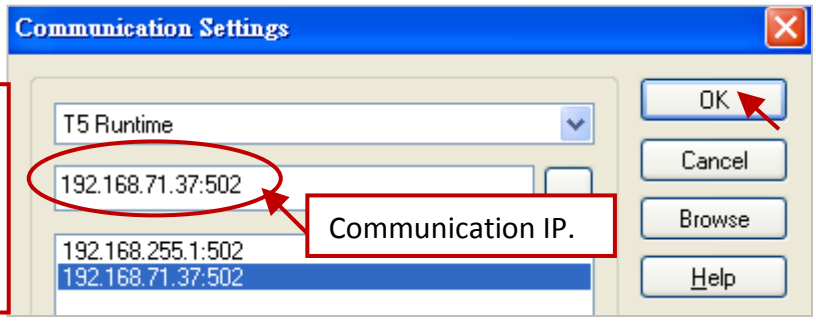
Note: If the user wants to set the timeout value (default: 3 seconds), see Section 2.3.5. (E.g. Set the IP to “192.168.255.1:502**(10)**” which means the timeout is **10** seconds.

3. Recompile the "demo_RDN_2” project and then download it to the Active PAC (refer Section 2.3.4 and Section 2.3.5 for details). After that, the LAN1 of the Active PAC will be automatically set as the Active IP (i.e., "192.168.71.37" in this example program, refer Section 16.3.1) and the LAN1 of the Passive PAC will be set as the Active IP + 1 (i.e., "192.168.71.38") automatically.





4. Right now, Win-GRAF will show "Communication error" because the current Active PAC IP (e.g., 192.168.71.37) and the communication IP settings on the workbench (e.g., 192.168.255.1) are not on the same network domain. So, stop the connection and change the communication IP of this "demo_RDN_2” project to "192.168.71.37" (refer Section 2.3.5 “Communication Parameters") and then check if the PC’s IP is on the same network domain (e.g., "192.168.71.x"). Then, this project will always link to the Active PAC whenever the user wants to debug it or change it.

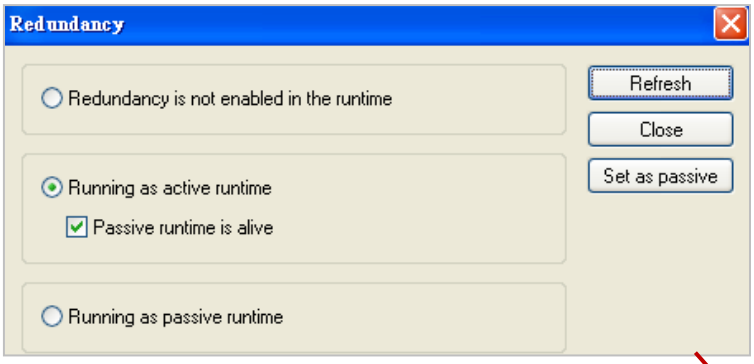


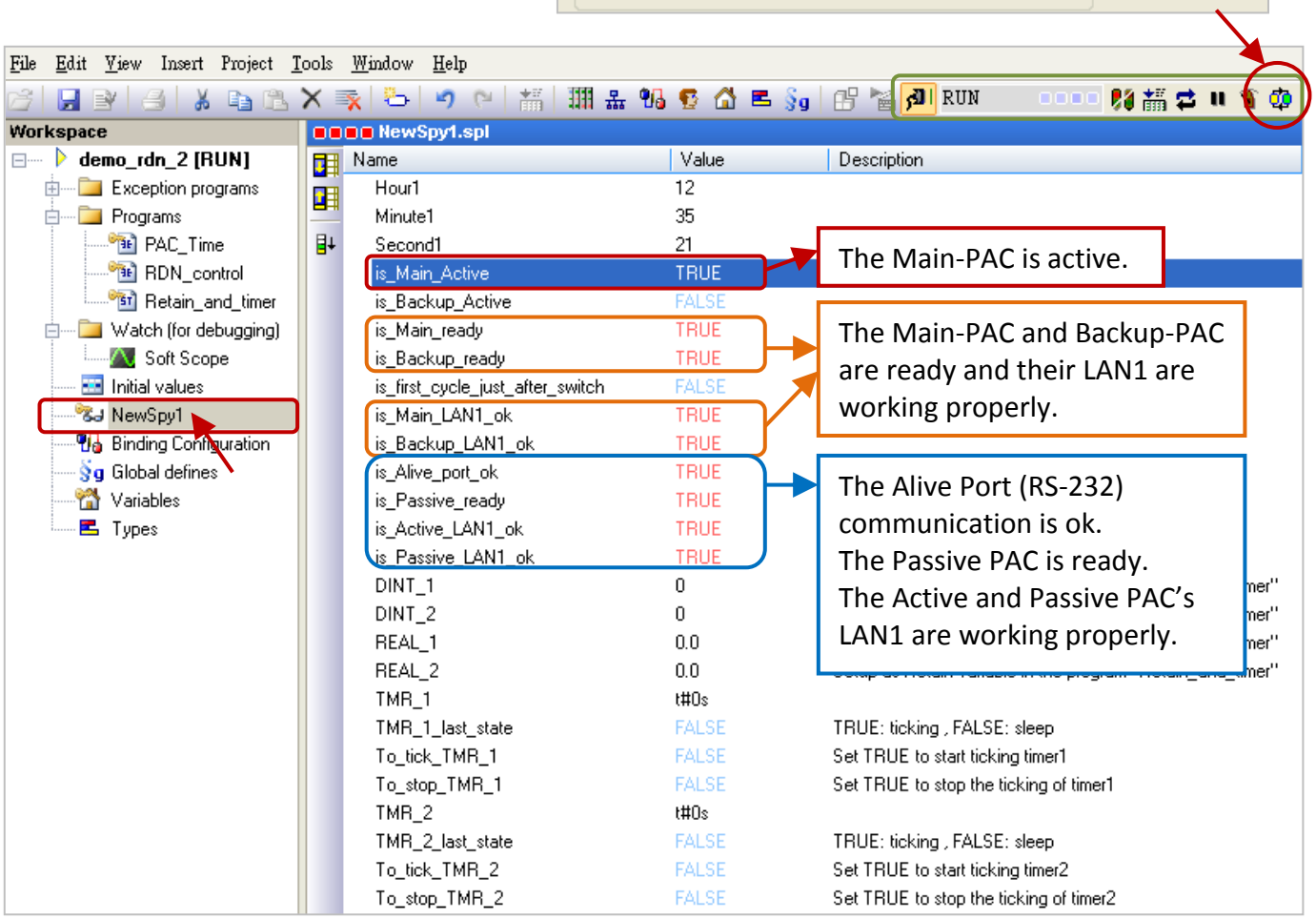
Note: If the user wants to set the timeout value (default: 3 seconds), see Section 2.3.5. (E.g. Set the IP to “192.168.71.37:502**(10)**” which means the timeout is **10** seconds.

5. After re-connecting with the PAC, click "NewSpy1" to open the spy list window and you can see the current Active PAC is the Main-PAC.

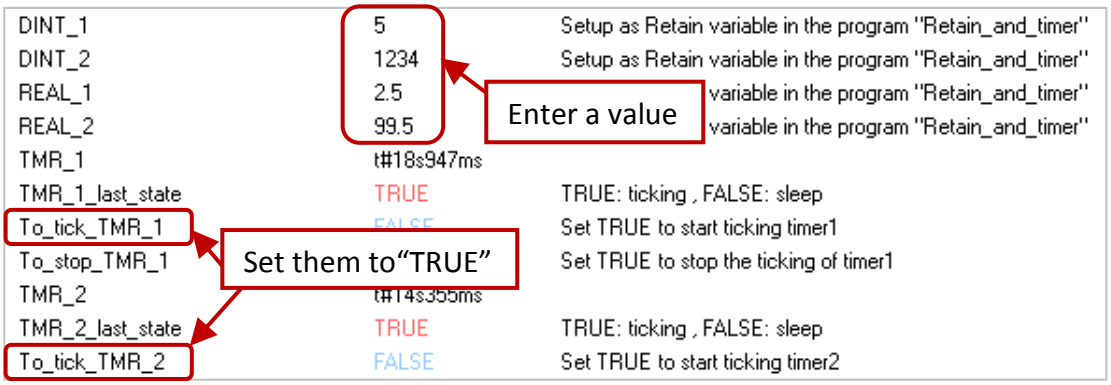
Note:

Don't switch control-right if the Passive PAC is not ready (i.e., "is_Passive_ready" = FALSE) because it cannot receive the redundant data from the Active PAC. The user can click the "Redundancy" button () to switch control-right for testing.

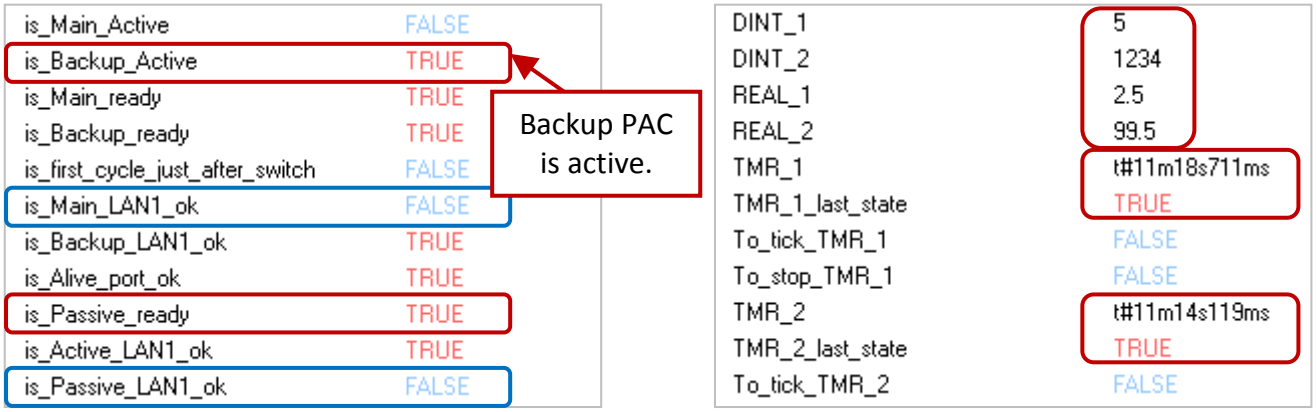




6. Enter values for "DINT_1", "DINT_2", "REAL_1" and "REAL _1" variables and then set the "To_tick_TMR_1" and "To_tick_TMR_2" as TRUE (it will auto reset to FALSE) to start the "TMR_1" and "TMR_2" ticking. Now, the status of TIMER will change from FALSE to TRUE.

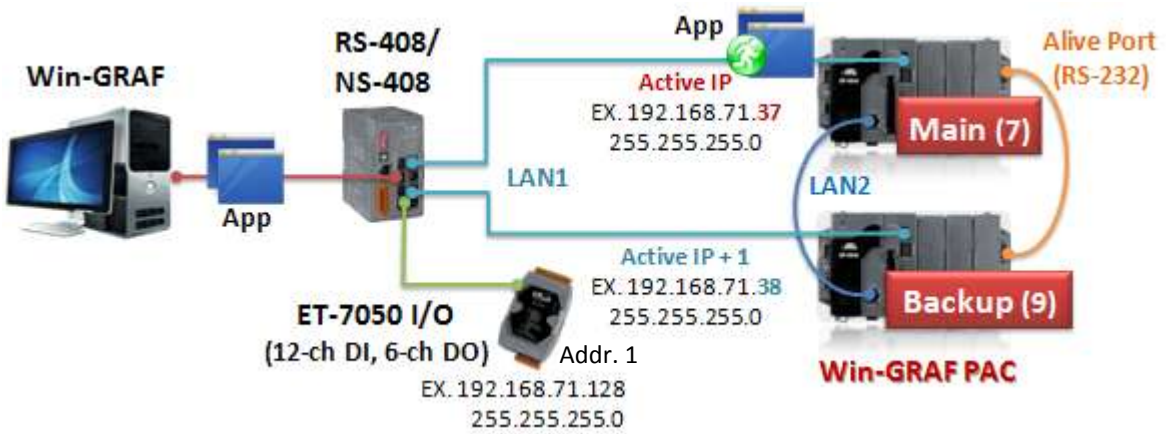


7. Make sure the Passive PAC is ready (i.e., is_Passive_Ready is TRUE), remove the LAN1 cable of the Main-PAC or turn off and on (restart) the Main-PAC. Wait for a short time (refer the “RDN_control” program), the Main-PAC will automatically reboot and give control-right to the Backup-PAC, Now. Then, the Active PAC belongs to the Backup-PAC and all the values you set before still exist and the Timer is still ticking.



(**Note:** After that, plug in the LAN1 cable of the Main-PAC again. Later, both the "is_Main_LAN1" and the "is_Passive_LAN1" status will become "TRUE".)

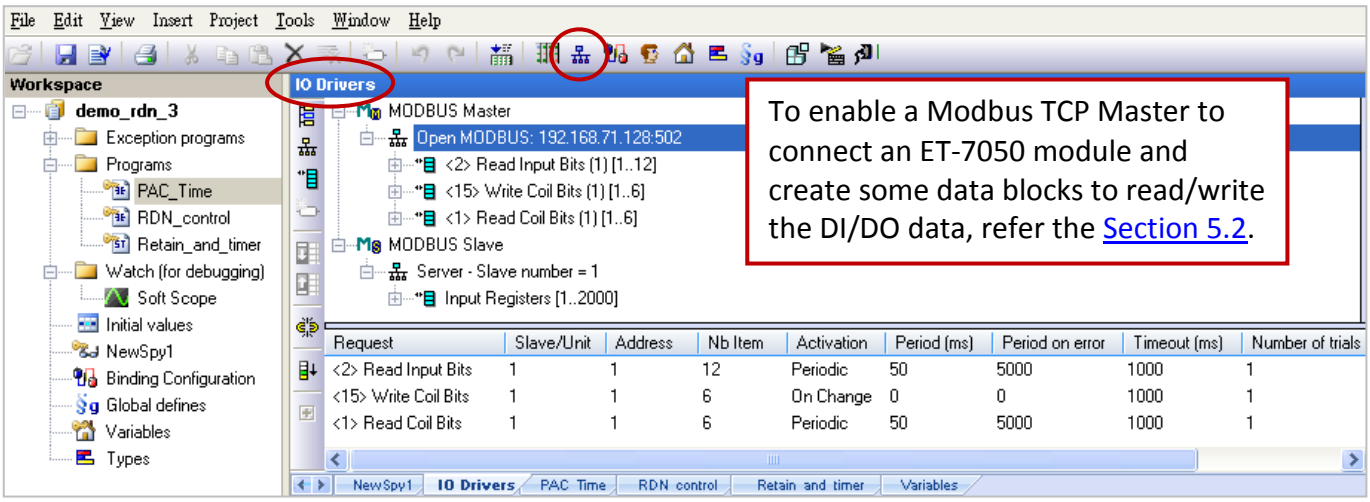
demo_RDN_3:
Two XP-8xx8-CE6 PACs, using their LAN1 to connect a ET-7050 (Modbus TCP I/O module) through the Ethernet switch.



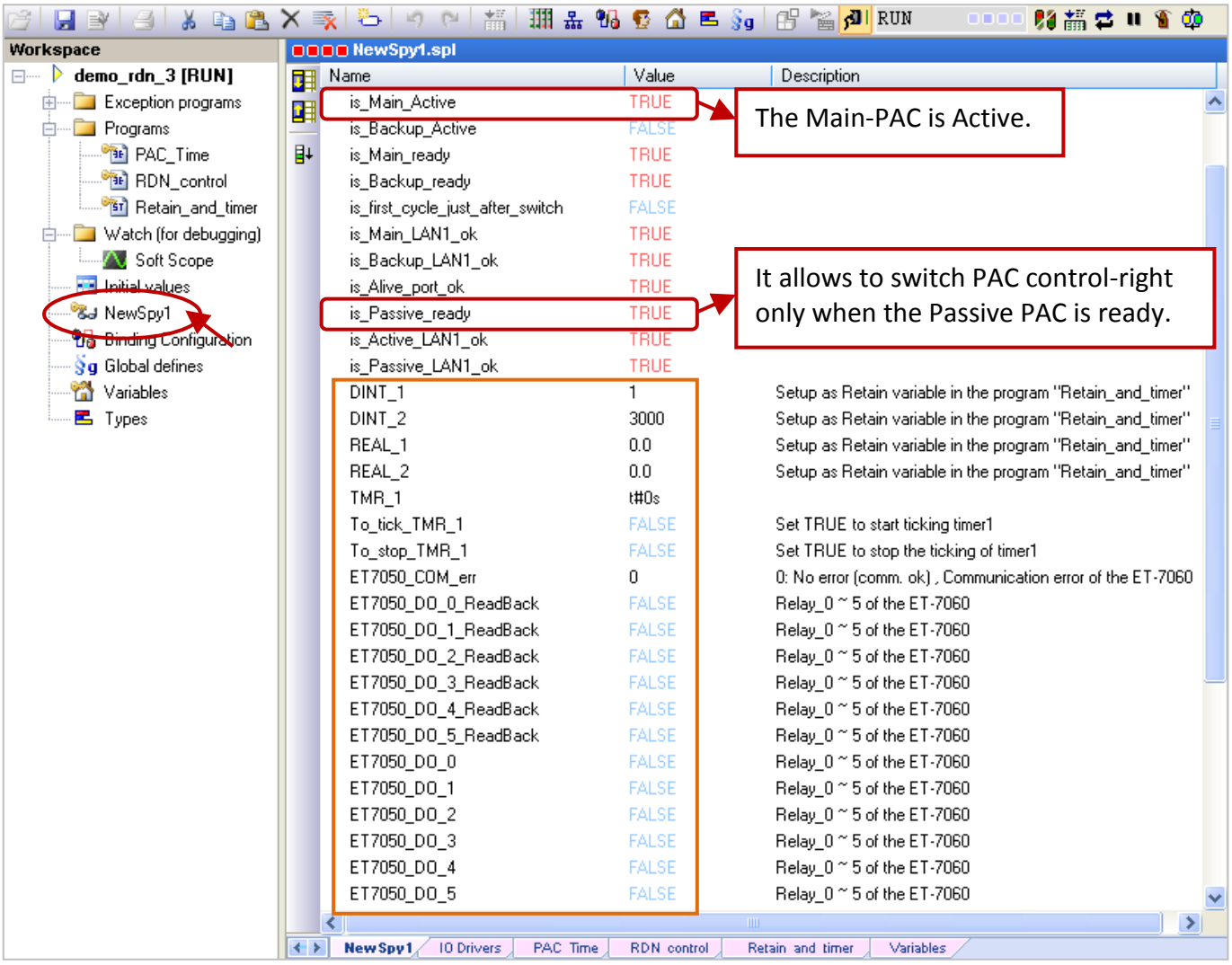
1. Refer the ET-7000 manual to set up the IP address and required settings (refer Section 5.2.1).
   Manual: http://ftp.icpdas.com/pub/cd/6000cd/napdos/et7000_et7200/document/

2. Open and download the "demo_RDN_3" project. Before downloading, set the communication IP (refer Section 2.3.5 “Communication Parameters") to the current LAN1 IP of the Active PAC (refer Section 16.4 – Test demo programs - step 2 to 4).

In the "I/O Drivers" window, here we enable a Modbus TCP Master to connect an ET-7050 module (Modbus TCP Slave, Addr. = 1) and create some data blocks to read/write the DI and DO data (refer Section 5.2). Users can open this "demo_RDN_3" project for more details.

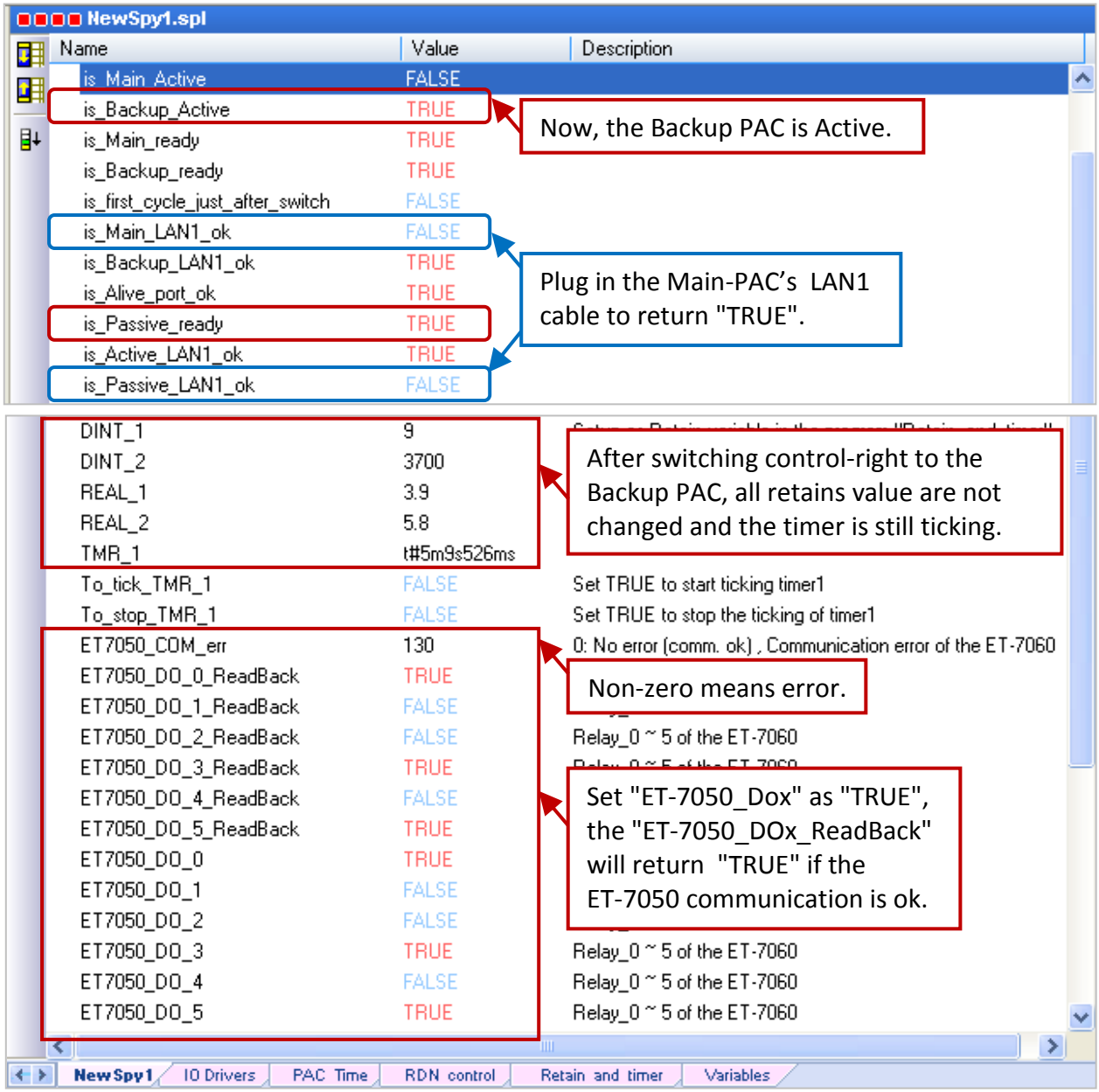


3. Click "NewSpy1" to open the spy list, now the Main-PAC is Active.

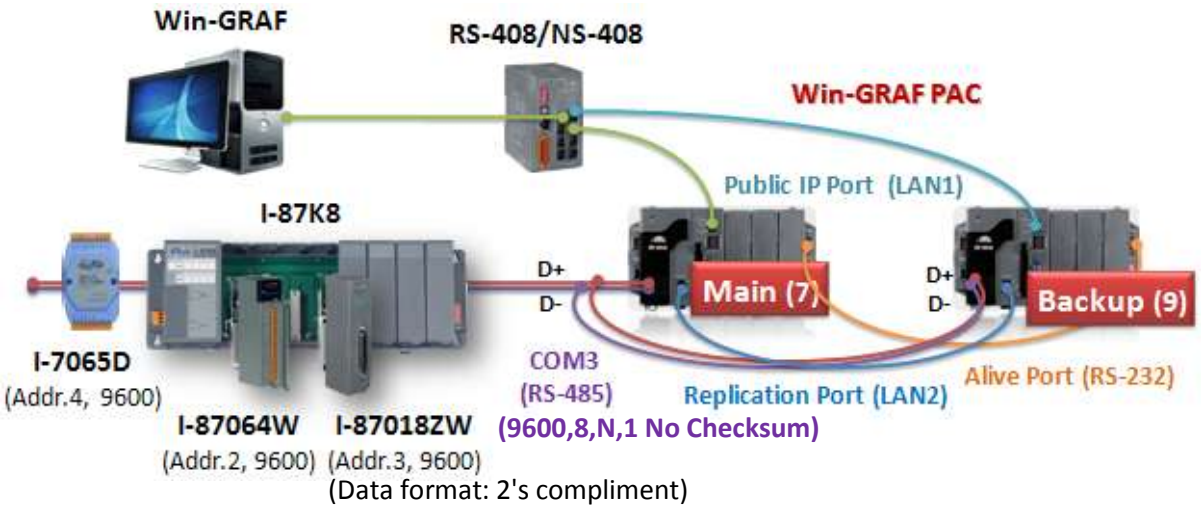


4. Enter some values for these retain variables (DINT_1, DINT_2, REAL_1 and REAL_2) and set "To_tick_TMR_1" as "TRUE" to start Timer. If set "ET-7050_Dox" as " TRUE", the "ET-7050_DOx_ReadBack" will return "TRUE". If disconnect the Ethernet cable from the ET-7050 module, the "ET-7050_COM_error" will return a non-zero value that means communication error.

5. Make sure the Passive PAC is ready (i.e., "is_Passive_Ready" is "TRUE"), remove the LAN1 cable from the Main-PAC or turn it off and on (restart), and wait a short time (refer the “RDN_control” program), the Main-PAC will automatically reboot and give control-right to the Backup-PAC.



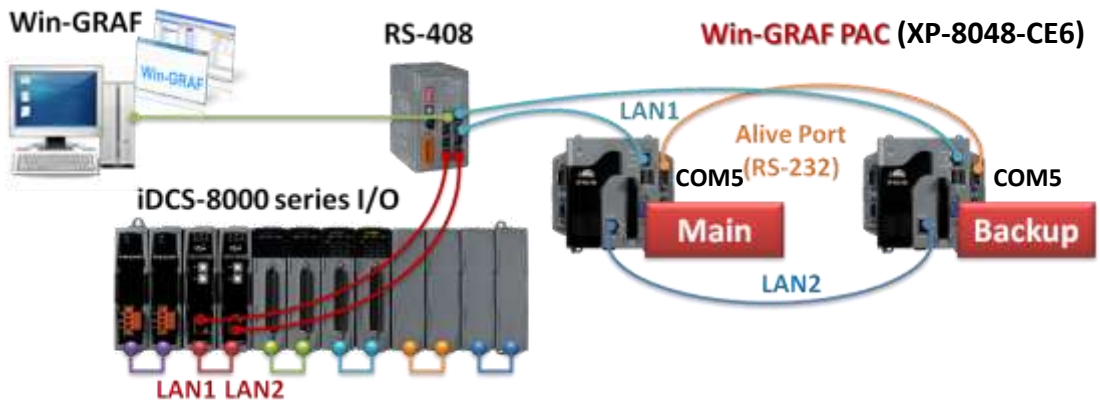
demo_RDN_1: Two XP-8xx8-CE6 PACs, using their COM3 to connect three DCON I/O modules.



In the "demo_RDN_1" demo project (as the figure above), before linking "I-7000" and "I-87KW" DCON remote I/O modules, users need to configure each of I/O modules by using "DCON Utility". Refer the description in Chapter 8 and visit the "DCON Utility" web page to download the software and user manual: www.icpdas.com/products/dcon/introduction.htm. Please restore and open this demo project for more details and refer Section 16.3.1 for "I/O Board" settings.

16.4.2 Test the "demo_RDN_4" Project

Two XP-8xx8-CE6 PACs, using their LAN1 to connect an iDCS-8830 (Both PAC and I/O are redundant) through one Ethernet switch.



The following table lists all used devices in this project:

Products	Quantity	Products	Quantity
XP-8048-CE6	2	DN-DI-32DW	1
RS-408	1	DN-DO-16DR-A	1
iDCS-8830	1	DN-DO-16DR-B	1
F-8040	2	CA-37**10**AM (1M Cable) or CA-37**20**AM/**30**AM/**50**AM/**100**AM	4
F-8041	2		

Hardware Wiring and Software Setting:

You can refer Section 16.1 and Section 16.2 to view the Win-GRAF redundant system (4) and the wiring way of two XP-8048-CE6 PACs. This iDCS-8830 is installed with the following I/O modules (from left to right). First, download and install the related utility and user manual.

iDCS-8000 Utility: http://ftp.icpdas.com/pub/cd/idcs-8000/utility/ (Software Installation: Ch2.2)

MiniOS7 Utility: http://ftp.icpdas.com/pub/cd/8000cd/napdos/minios7/utility/

iDCS-8000 User Manual/Website:

http://ftp.icpdas.com/pub/cd/idcs-8000/usersmanual/

http://www.icpdas.com/root/product/solutions/remote_io/dcs_redundancy_io/idcs_introduction.html

FPM-D2440 * 2 : Power module1, Power module2 (for power input, 24V).

FCM-MTCP * 2 :

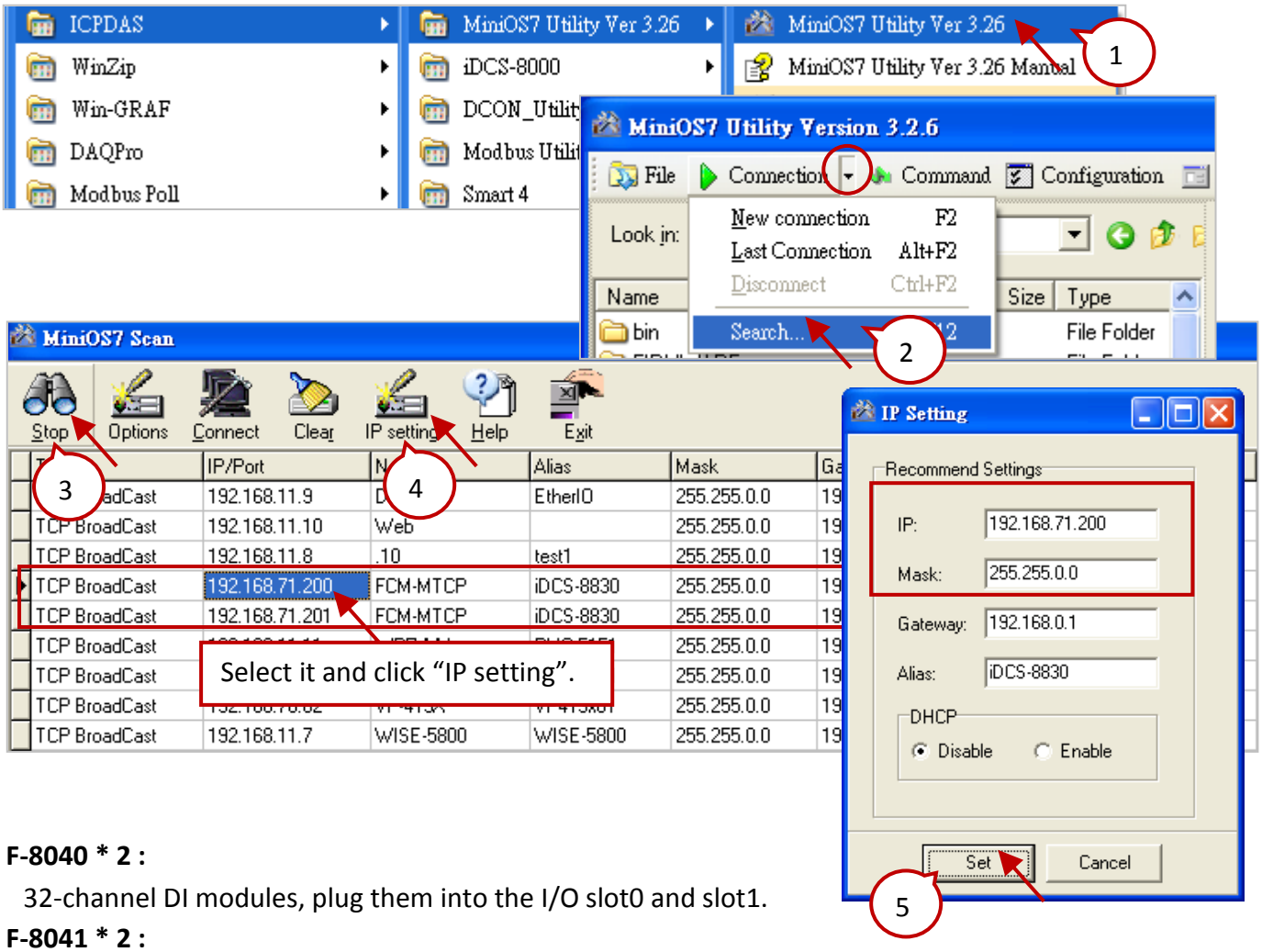
MCU1 (First, rotate the SW2 to "C" and the SW1 to "8", then set up the IP address to 192.168.71.**200**).

MCU2 (First rotate the SW2 to "C", the SW1 to "9", and then set the IP address to 192.168.71.**201**).

The SW2/SW1 means the fourth IP address of the MCU (Main Control Unit). ($C8_{16}$ = 200; $C9_{16}$ = 201)

Refer the iDCS-8000 user manual (Ch2.3 or the figure below) to set IP addresses by using the "MiniOS7 Utility".

Run the “MiniOS7 Utility”, click “Search” to search current IP addresses of the iDCS-8830 (i.e., MCU1/2), and click “Stop” afterwards. Then, change these IP addresses to “192.168.71.200” and “192.168.71.201”, change the Mask to “255.255.**0.0**” and click “Set”. Finally, close the “MiniOS7 Utility”.



F-8040 * 2 :

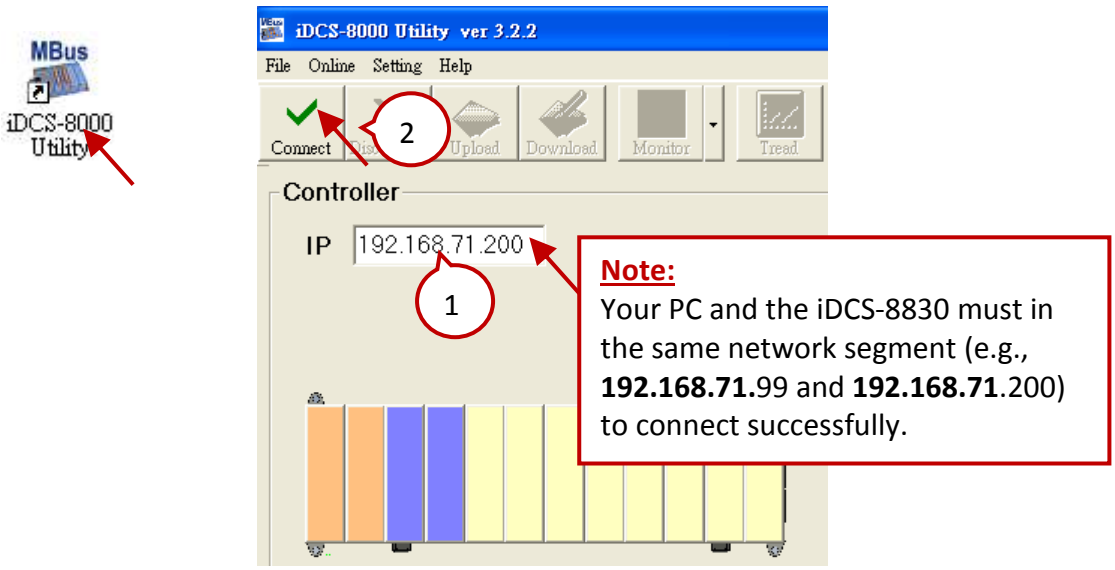
32-channel DI modules, plug them into the I/O slot0 and slot1.

F-8041 * 2 :

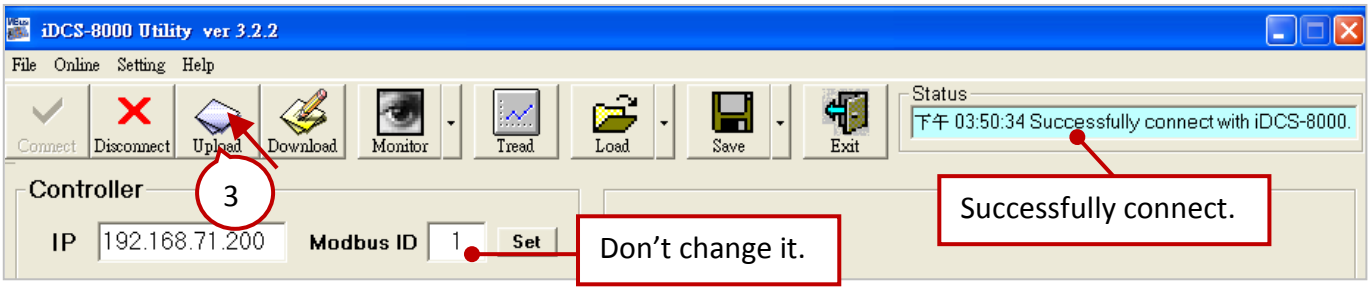
32-channel DO modules, plug them into the I/O slot2 and slot3.

Refer the iDCS-8000 user manual (Ch2.3 or the figure below) to configure the I/O module by using the iDCS-8000 Utility.

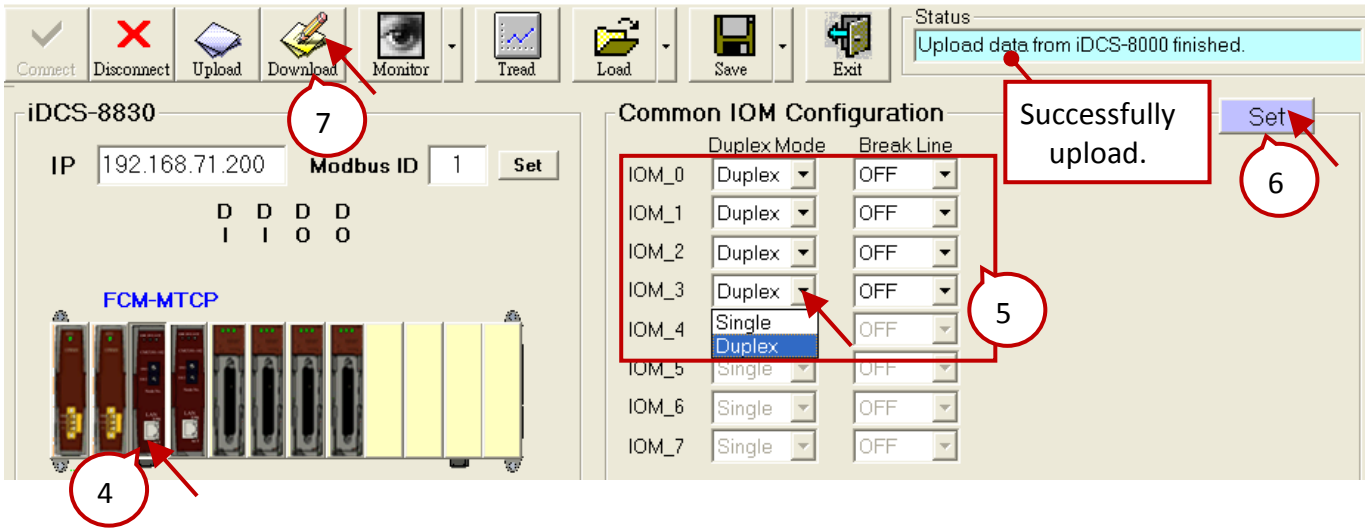
Run the "iDCS-8000 Utility" and fill in the IP address of the iDCS-8830 to connect.



Click "Upload" to upload the current settings of I/O modules that plugged into the iDCS-8830.



Click the 1st FCM-MTCP (MCU1) and set "IOM_0 ~ 3" (F-8040/F-8041) as "Duplex" Mode (If the F-8040/ F-8041 module isn't connected with the following termination boards now, set the "Break Line" as "OFF") and click "Set". Then, click "Download" to download the above settings to the iDCS-8830. Finally, close the "iDCS-8000 Utility".



If you have connected the following termination boards, set the "Break Line" as "ON" (the figure above).

DN-DI-32DW * 1 : 32-channel DI termination board.
Connect it to the F-8040 DI module (slot0, slot1).

DN-DO-16DR-A * 1 : 16-channel DO termination board (for channel 0 ~ 15).

DN-DO-16DR-B * 1 : 16-channel DO termination board (for channel 16 ~ 31).
Connect the DN-DO-16DR-A (CN1, CN2) to the F-8041 DO module (slot2, slot3), and then connect the DN-DO-16DR-A (CN3) to the DN-DO-16DR-B (CN1).

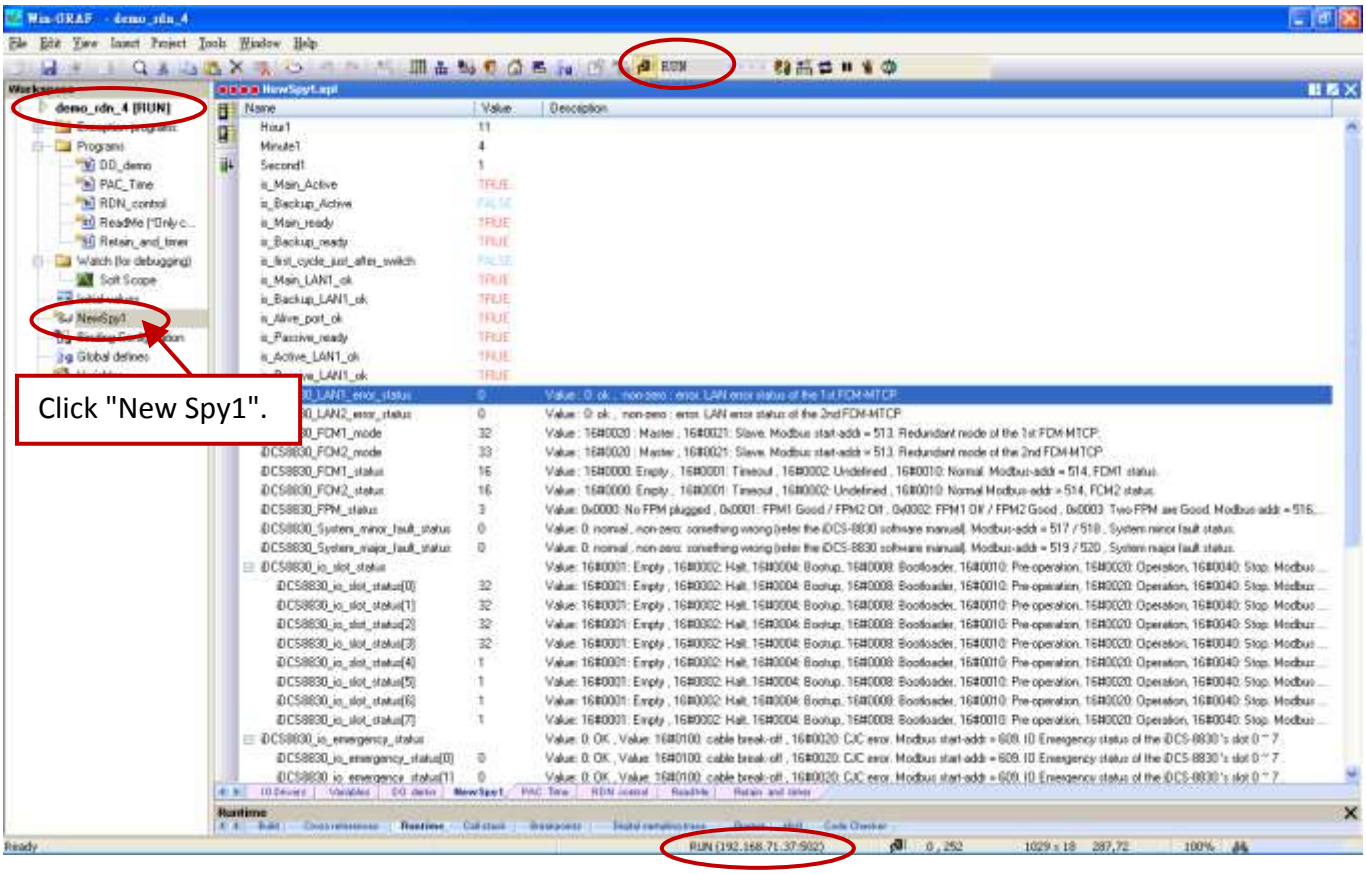
CA-3710AM * 4 : 1m 37-pin Male-Female D-sub Cable.
It used to connect the I/O module and the termination board.

Start testing:

First, the user must go to Section 16.4.1 to know how to set the communication IP address, download the Win-GRAF project ("demo_RDN_4") and test the redundant PAC system. The following descriptions will focus on the iDCS-8830. Before starting the test, make sure all devices are connected properly and then run the Win-GRAF Workbench to connect to the PAC.

Note: Make sure both of IP addresses of your PC and PAC are on the same network segment.
(e.g., PAC IP = **192.168.71**.37, Mask = **255.255.255**.0, set your PC IP = **192.168.71**.x)

After connecting to the PAC, click "New Spy1" to open the spy list.



iDCS-8830's Communication Status:

Name	iDCS8830_LAN1_error_status	iDCS8830_LAN2_error_status
Test Steps	1. Unplug (or plug in) the LAN cable of the 1st FCM-MTCP. Value: Non-0 (or 0). 2. Unplug (or plug in) the LAN cable of the 2nd FCM-MTCP. Value: Non-0 (or 0).	
Return Value	0 : Communication OK ; Non-0 : Communication Error	

iDCS8830_LAN1_error_status	130	Value : 0: ok , non-zero : error. LAN error status of the 1st FCM-MTCP.
iDCS8830_LAN2_error_status	0	Value : 0: ok , non-zero : error. LAN error status of the 2nd FCM-MTCP.

iDCS-8830's Power Status:

Name	iDCS8830_FPM_status
Test Steps	1. Unplug (or plug in) the power cable of the 1st FCM-MTCP. Value: 2 (or 3). 2. Unplug (or plug in) the power cable of the 2nd FCM-MTCP. Value: 1 (or 3).
Return Value	0 : No FPM is plugged. 1 : FPM1 Off ; FPM2 Good. 2 : FPM1 Good ; FPM2 Off. 3 : Both of these FPM are plugged.

iDCS8830_LAN1_error_status	0	Value : 0: ok , non-zero : error. LAN error status of the 1st FCM-MTCP.
iDCS8830_LAN2_error_status	0	Value : 0: ok , non-zero : error. LAN error status of the 2nd FCM-MTCP.
iDCS8830_FCM1_mode	32	Value : 16#0020 : Master , 16#0021: Slave. Modbus start-addr = 513. Redundant mode of the 1st FCM-
iDCS8830_FCM2_mode	33	Value : 16#0020 : Master , 16#0021: Slave. Modbus start-addr = 513. Redundant mode of the 2nd FCM
iDCS8830_FCM1_status	16	Value : 16#0000: Empty , 16#0001: Timeout , 16#0002: Undefined , 16#0010: Normal. Modbus-addr =
iDCS8830_FCM2_status	16	Value : 16#0000: Empty , 16#0001: Timeout , 16#0002: Undefined , 16#0010: Normal Modbus-addr = 5
iDCS8830_FPM_status	2	Value: 0x0000: No FPM plugged , 0x0001: FPM1 Good / FPM2 Off , 0x0002: FPM1 Off / FPM2 Good ,

FCM-MTCP's Redundant Mode and Status:

Name	iDCS8830_FCM1_mode iDCS8830_FCM1_status	iDCS8830_FCM2_mode iDCS8830_FCM2_status
Test Step	Unplug the 1st FCM-MTCP module (FCM1).	
	iDCS8830_LAN1_error_status 130 Value : 0: ok , non-zero : error. LAN error status of the 1st FCM-MTCP. iDCS8830_LAN2_error_status 0 Value : 0: ok , non-zero : error. LAN error status of the 2nd FCM-MTCP. iDCS8830_FCM1_mode 32 Value : 16#0020 : Master , 16#0021: Slave. Modbus start-addr = 513. R iDCS8830_FCM2_mode 32 Value : 16#0020 : Master , 16#0021: Slave. Modbus start-addr = 513. R iDCS8830_FCM1_status 1 Value : 16#0000: Empty , 16#0001: Timeout , 16#0002: Undefined , 16 iDCS8830_FCM2_status 16 Value : 16#0000: Empty , 16#0001: Timeout , 16#0002: Undefined , 16	
Description	iDCS8830_LAN1_error_status = 130 It means the FCM1 communication error. iDCS8830_FCM1_mode = 32 The redundant mode still displayed as "32" (Master) because the FCM1 is unplug. iDCS8830_FCM2_mode = 32 It means the FCM2 is taking over now and its redundant mode is "Master" (32). iDCS8830_FCM1_status = 1 means that the FCM1 is timeout.	
Return Value	iDCS8830_FCM1_mode / iDCS8830_FCM2_mode 32 : Master 33: Slave iDCS8830_FCM1_status / iDCS8830_FCM2_status 0: Empty 1: Timeout 2: Undefined 16: Normal	

Plug in the FCM1, it will change to "Slave" mode (33), and becomes a normal status (16).

iDCS8830_LAN1_error_status	0	Value : 0: ok , non-zero : error. LAN error status of the 1st FCM-MTCP.
iDCS8830_LAN2_error_status	0	Value : 0: ok , non-zero : error. LAN error status of the 2nd FCM-MTCP.
iDCS8830_FCM1_mode	33	Value : 16#0020 : Master , 16#0021: Slave. Modbus start-addr = 513. R
iDCS8830_FCM2_mode	32	Value : 16#0020 : Master , 16#0021: Slave. Modbus start-addr = 513. R
iDCS8830_FCM1_status	16	Value : 16#0000: Empty , 16#0001: Timeout , 16#0002: Undefined , 16
iDCS8830_FCM2_status	16	Value : 16#0000: Empty , 16#0001: Timeout , 16#0002: Undefined , 16

You can try to unplug the FCM2, and then the FCM1 will become the Master (32) and take over.

iDCS8830_LAN1_error_status	0	Value : 0: ok , non-zero : error. LAN error status of the 1st FCM-MTCP.
iDCS8830_LAN2_error_status	130	Value : 0: ok , non-zero : error. LAN error status of the 2nd FCM-MTCP.
iDCS8830_FCM1_mode	32	Value : 16#0020 : Master , 16#0021: Slave. Modbus start-addr = 513. R
iDCS8830_FCM2_mode	33	Value : 16#0020 : Master , 16#0021: Slave. Modbus start-addr = 513. R
iDCS8830_FCM1_status	16	Value : 16#0000: Empty , 16#0001: Timeout , 16#0002: Undefined , 16
iDCS8830_FCM2_status	1	Value : 16#0000: Empty , 16#0001: Timeout , 16#0002: Undefined , 16

Plug in the FCM2 afterward.

iDCS8830_LAN1_error_status	0	Value : 0: ok , non-zero : error. LAN error status of the 1st FCM-MTCP.
iDCS8830_LAN2_error_status	0	Value : 0: ok , non-zero : error. LAN error status of the 2nd FCM-MTCP.
iDCS8830_FCM1_mode	32	Value : 16#0020 : Master , 16#0021: Slave. Modbus start-addr = 513. R
iDCS8830_FCM2_mode	33	Value : 16#0020 : Master , 16#0021: Slave. Modbus start-addr = 513. R
iDCS8830_FCM1_status	16	Value : 16#0000: Empty , 16#0001: Timeout , 16#0002: Undefined , 16
iDCS8830_FCM2_status	16	Value : 16#0000: Empty , 16#0001: Timeout , 16#0002: Undefined , 16

The Status of I/O Slot and I/O Emergency:

Name	iDCS8830_io_slot_status	iDCS8830_io_emergency_status
Test step	Unplug the cable between the 1st F-804**0** (slot 0) and the terminal board DN-DI-32DW.	
	iDCS8830_io_slot_status — Value: 16#0001: Empty , 16#0002: Halt, 16#0004: Bootup, 16#0008 iDCS8830_io_slot_status[0] 64 — Value: 16#0001: Empty , 16#0002: Halt, 16#0004: Bootup, 16#0008 iDCS8830_io_slot_status[1] 32 — Value: 16#0001: Empty , 16#0002: Halt, 16#0004: Bootup, 16#0008 iDCS8830_io_slot_status[2] 32 — Value: 16#0001: Empty , 16#0002: Halt, 16#0004: Bootup, 16#0008 iDCS8830_io_slot_status[3] 32 — Value: 16#0001: Empty , 16#0002: Halt, 16#0004: Bootup, 16#0008 iDCS8830_io_slot_status[4] 1 — Value: 16#0001: Empty , 16#0002: Halt, 16#0004: Bootup, 16#0008 iDCS8830_io_slot_status[5] 1 — Value: 16#0001: Empty , 16#0002: Halt, 16#0004: Bootup, 16#0008 iDCS8830_io_slot_status[6] 1 — Value: 16#0001: Empty , 16#0002: Halt, 16#0004: Bootup, 16#0008 iDCS8830_io_slot_status[7] 1 iDCS8830_io_emergency_status iDCS8830_io_emergency_status[0] 256 iDCS8830_io_emergency_status[1] 0 iDCS8830_io_emergency_status[2] 0 iDCS8830_io_emergency_status[3] 0 **Note:** **The "iDCS8830_io_emergency_status[x]" works only when you set the "IOM_x" as "Duplex" Mode.**	
Description	iDCS8830_io_slot_status[0] = 64, which means this module (slot0) is stop working. iDCS8830_io_emergency_status[0] = 256 It means the cable which plugged into this module (slot0) is disconnected. Plug in this cable and then you can try to unplug others cable for testing.	
Return Value	iDCS8830_io_slot_status[x] 1: Empty 2: Halt 4: Bootup 8: Bootloader 16: Pre-operation 32: Operation 64: Stop iDCS8830_io_emergency_status[x] 0 : Normal 32: CJC Error 256: Cable Break-off	

Redundant DO module:

Name	iDCS8830_slot23_F8041_DO
Test step	Plug in the 1st F-804**1** DO module (slot2).
	iDCS8830_io_slot_status — Value: 16#0001: Empty , 16#0002: Halt, iDCS8830_io_slot_status[0] 32 — Value: 16#0001: Empty , 16#0002: Halt, iDCS8830_io_slot_status[1] 32 — Value: 16#0001: Empty , 16#0002: Halt, iDCS8830_io_slot_status[2] 2 — Value: 16#0001: Empty , 16#0002: Halt, iDCS8830_io_slot_status[3] 32 — Value: 16#0001: Empty , 16#0002: Halt, iDCS8830_slot23_F8041_DO — F-8041 Digital Output in the slot 2 - 3 (Duplex mode) of iDCS-8830 iDCS8830_slot23_F8041_DO[0] TRUE — F-8041 Digital Output in the slot 2 - 3 (Duplex mode) of iDCS-8830 iDCS8830_slot23_F8041_DO[1] TRUE — F-8041 Digital Output in the slot 2 - 3 (Duplex mode) of iDCS-8830 iDCS8830_slot23_F8041_DO[2] TRUE — F-8041 Digital Output in the slot 2 - 3 (Duplex mode) of iDCS-8830 iDCS8830_slot23_F8041_DO[3] TRUE — F-8041 Digital Output in the slot 2 - 3 (Duplex mode) of iDCS-8830 iDCS8830_slot23_F8041_DO[4] TRUE — F-8041 Digital Output in the slot 2 - 3 (Duplex mode) of iDCS-8830 iDCS8830_slot23_F8041_DO[5] TRUE — F-8041 Digital Output in the slot 2 - 3 (Duplex mode) of iDCS-8830 iDCS8830_slot23_F8041_DO[6] TRUE — F-8041 Digital Output in the slot 2 - 3 (Duplex mode) of iDCS-8830 iDCS8830_slot23_F8041_DO[7] TRUE — F-8041 Digital Output in the slot 2 - 3 (Duplex mode) of iDCS-8830
Description	At first, the LED indicator (called DO0 ~ 7) of the 1st F-8041 module (slot2) will light up sequentially. When you unplug this module, the 2nd F-8041 module (slot3) will take over and do the same thing. Then, you can see the status value of the iDCS8830_io_slot_status[2] is "2" which means this module (slot2) is halted. (Plug in this module again.)

16.5 What Kinds of Data Can be Automatically Backed up to the Passive PAC?

In the Win-GRAF redundant system, not all of the data in the Active PAC can be automatically backed up to the Passive PAC.

What Can be Backed Up Automatically:

1. The user's Win-GRAF applications.
2. The execution step of programs.
3. Value of variables.
4. Private data of Function Block instance.
5. The PAC's RTC (Real Time Clock) time.
6. Retain memory.
7. Schedule-control configuration (refer Chapter 17).

What are NOT Backed Up Automatically?

The following are the most common items that cannot be automatically backed up to the Passive PAC.

1. The status of Timer variable (Ticking or Sleeping).
2. Files in the Active PAC (e.g., files located in the path "\system_disk" or "\Micro_SD", or files not belong to the Win-GRAF applications, such as C, VB.net, C#, and ELogger applications). These files cannot be backed up automatically. So all of them should be pre-installed in a spare (or repaired) PAC before mounting this PAC in the redundant system).
   (Exclusive of user designed Win-GRAF application and schedule-control configuration, which can be backed up automatically.)
3. If using the COM_OPEN() function to open the serial port, it will not be automatically opened again after switching PAC control-right.
4. The PAC's EEPROM memory cannot be backed up automatically.

All the items which unable to back up automatically, users can use the following similar procedures to deal with them. (Refer the "Retain_and_timer" program in the "demo_RDN_2" project)

```
if  is_first_cycle_just_after_switch  then

  (* Just in the cycle after switching. *)
  ……

end_if ;
```

Chapter 17 Schedule Control

Introduction:

All Win-GRAF WinCE series PAC support the Schedule-Control function. One PAC can control max. 10 Targets (devices) with specified schedule configurations. Each schedule Target (device) contains three variables to be controlled – one BOOL variable, one DINT variable and one REAL variable. ICP DAS provides a free software – "Schedule-Control Utility". User can use this software to edit /modify the schedule configurations easily in PC or in PAC.

Driver version of Win-GRAF PAC:

The Win-GRAF PAC supports Schedule-Control in the below driver version and the new version.

WinCE PAC	Win-GRAF PAC	Driver Version
ViewPAC	VP-x2x8-CE7	1.02
WP-8000	WP-8148, WP-8448, WP-8848	1.02
WP-8000-CE7	WP-8128-CE7, WP-8428-CE7, WP-8828-CE7	1.01
WP-5000	WP-5238-CE7	1.01
XP-8000-CE6	XP-8048-CE6, XP-8348-CE6, XP-8748-CE6	1.01

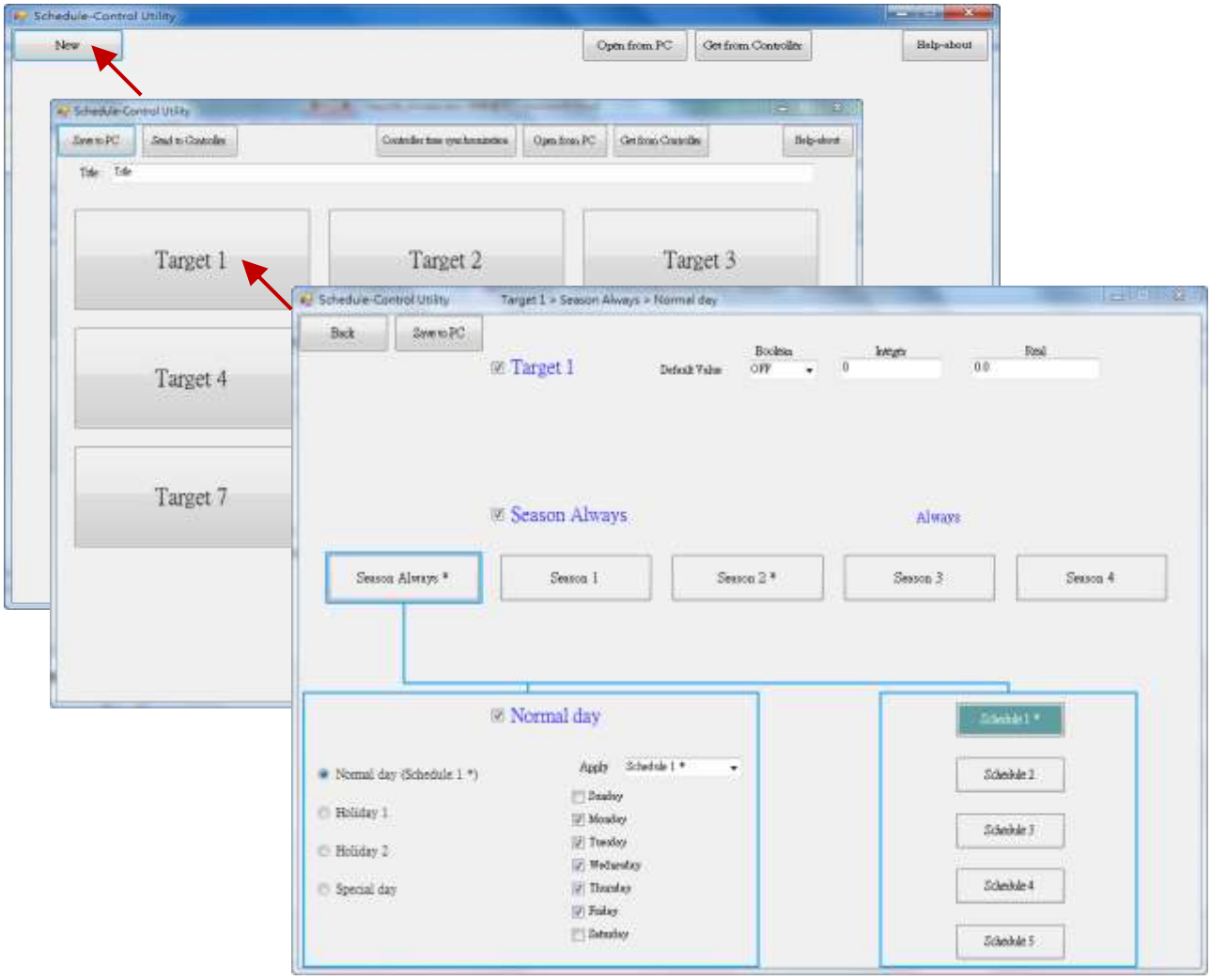
You may download newer Win-GRAF driver at
http://www.icpdas.com/root/product/solutions/softplc_based_on_pac/win-graf/download/win-graf-driver.html

17.1 Install the Schedule-Control Utility and Restore the Win-GRAF Demo Project

There is one Win-GRAF-PAC-CD in the Win-GRAF PAC package box. The Schedule-Control Utility file name is "**Schedule_in_PC.exe**" in the "CD:\napdos\Win-GRAF\Tools_Utility\" path.

Please copy this Schedule-Control Utility (Schedule_in_PC.exe and label_name.txt) to your PC. Recommend to copy it to the directory of your Win-GRAF project. For instance, copy it to "D:\Schedule-Control**Station1**\Schedule_in_PC.exe ", then run this "Schedule_in_PC.exe" file.

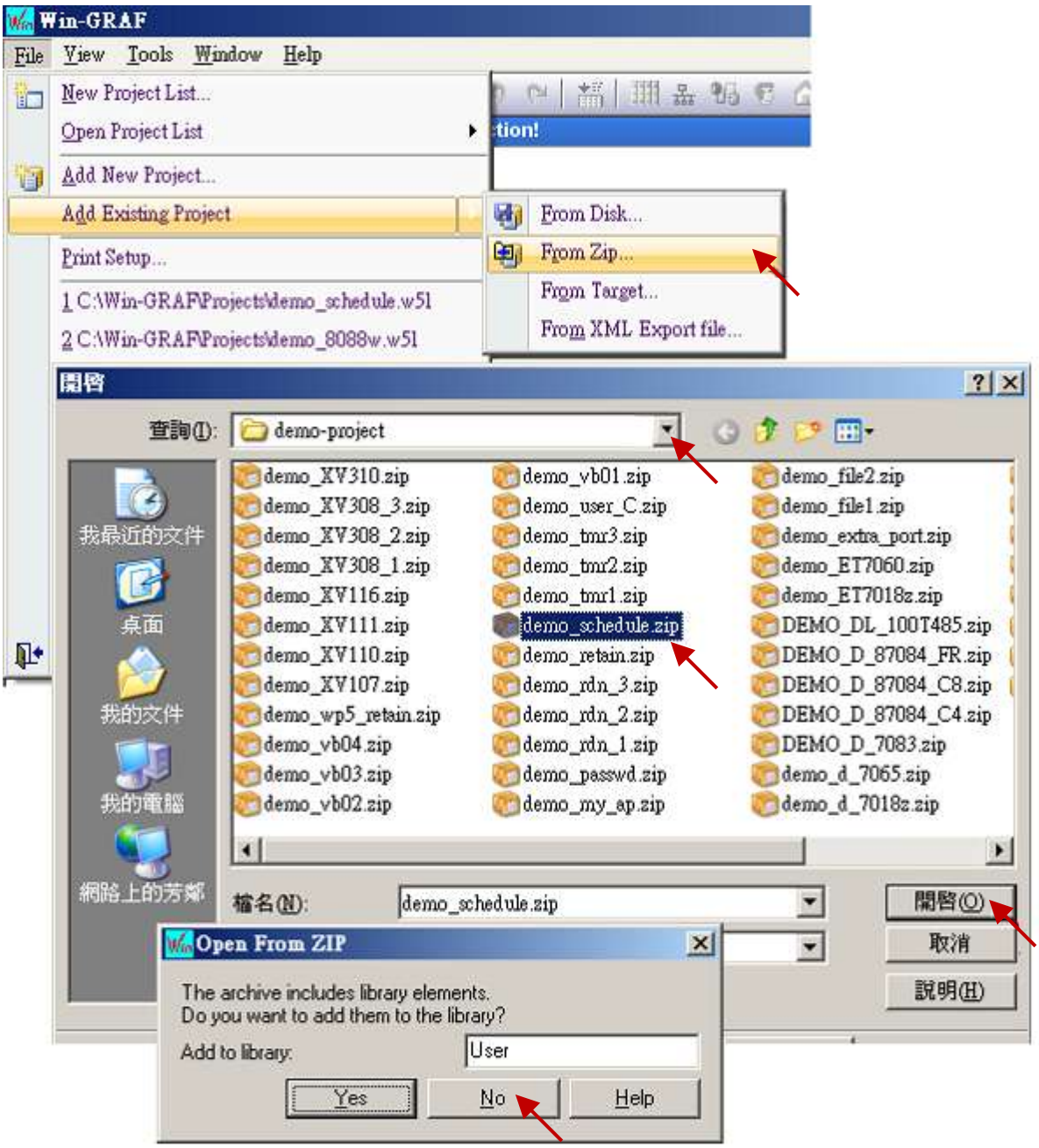
Then we will see the following windows (Click New > Target 1).



There is one another Schedule-control utility, however it is set up on the **PAC** (not on the PC). It is "**Schedule_in_PAC.exe**". You can find it in the "\System_Disk\Win-GRAF\" path of the Win-GRAF PAC.

Restore Win-GRAF demo project :

The Win-GRAF demo project for the Schedule-Control is “demo_schedule.zip”. It is in the “\napdos\Win-GRAF\demo-project\” path of the Win-GRAF-PAC-CD. Restore it to the PC / Win-GRAF workbench by the following way.

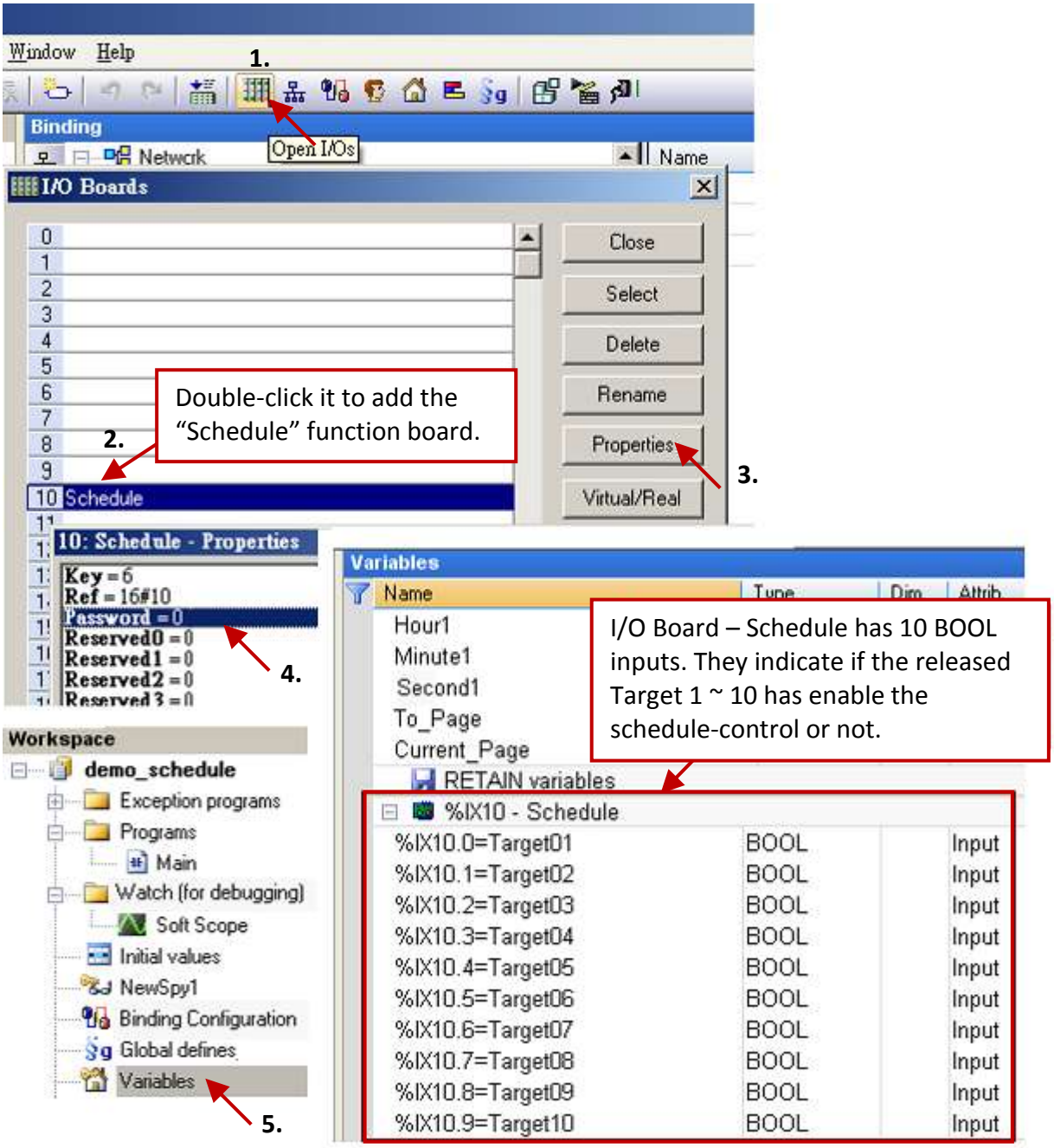


17.2 Introduction of the "demo_schedule" Project

This "demo_schedule" project shows the way to do schedule-control. Please prepare one Win-GRAF PAC (like VP-x238-CE7 or WP-8x48). One PAC can control schedules of max. 10 Targets (Target 1 to Target 10). Each Target contains one BOOL, one DINT and one REAL variable.

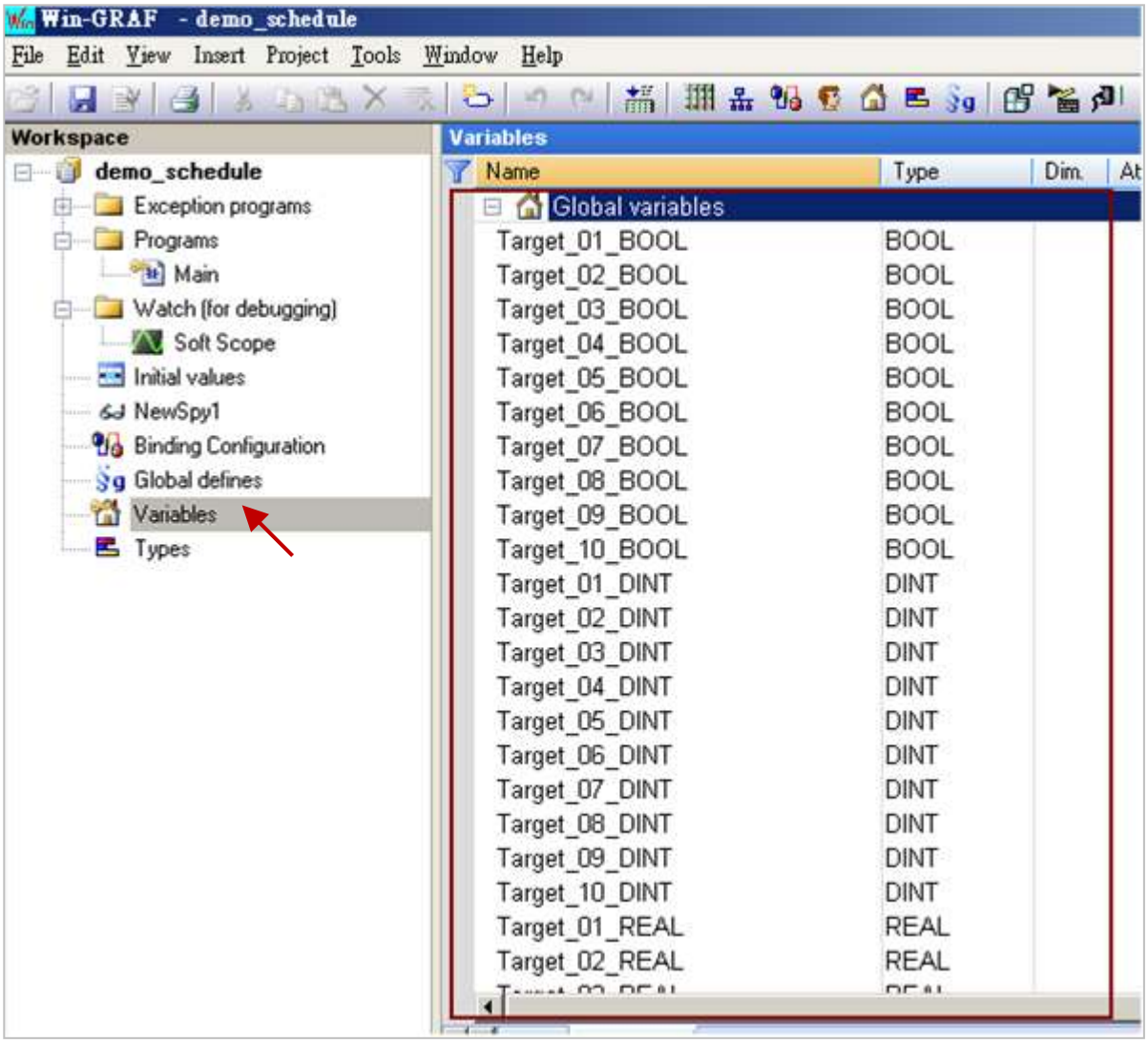
Settings in the "I/O boards" window :

To enable schedule-control in the Win-GRAF PAC, first click the "Open I/Os" to add one "Schedule" (add it in the slot number 8 or bigger number). There is a "Password" parameter in its "Properties" window. The "Password" is for the "Schedule-Control Utility" running in PC to identify the authorization when connecting the Win-GRAF PAC. It is set as 0 in this demo project. After adding the "Schedule" in the "I/O boards", we can find 10 BOOL input channels in the "variables" window. These 10 channels return the state of the schedule-control of the Target 1 to 10. TRUE means the Target has the schedule-control enabled by the "Schedule-Control utility". FALSE means not enabled.



Variable declaration:

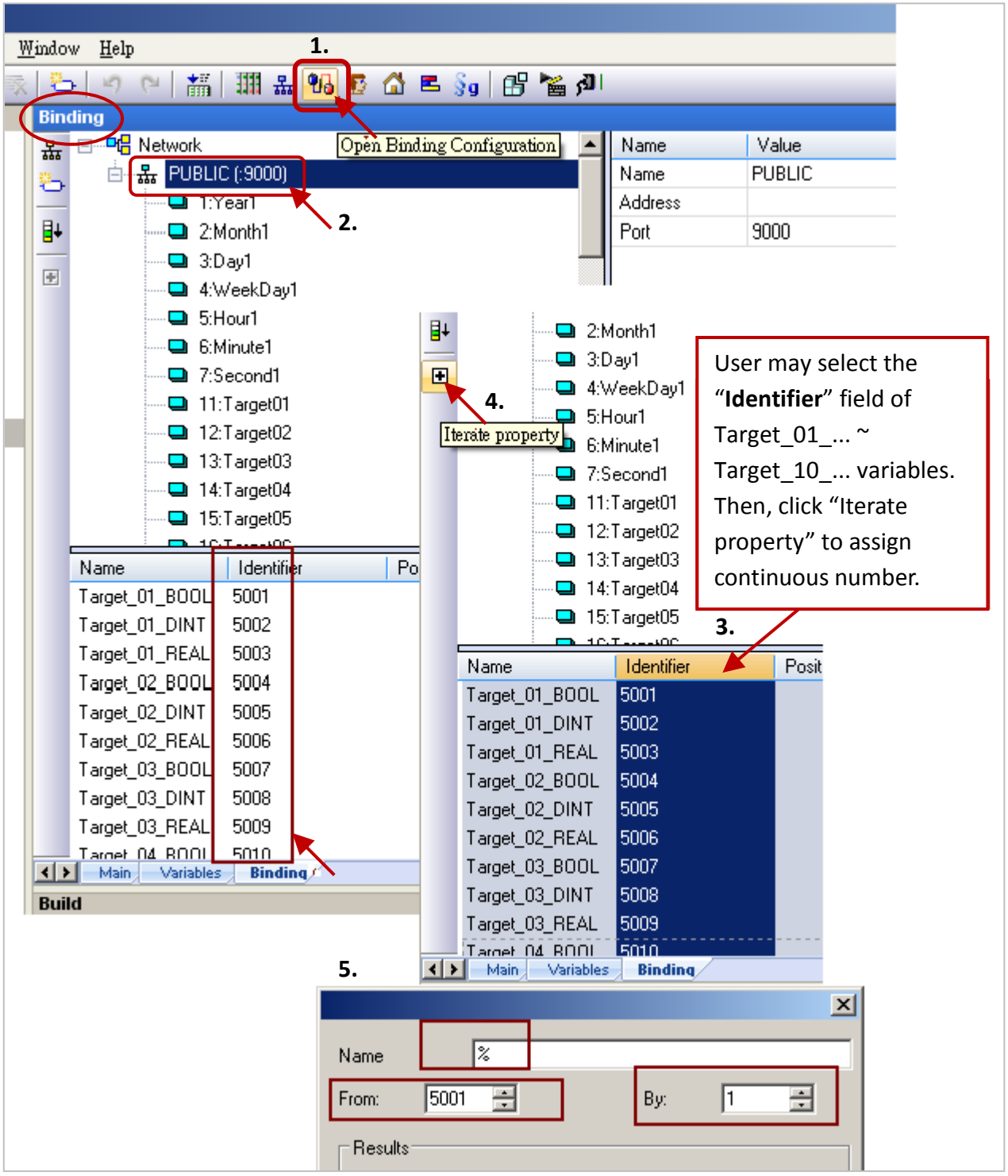
Click the “Variables” to view all variables in this demo project.



The above variables - "Target_01_BOOL ~ Target_10_BOOL” , “Target_01_DINT ~ Target_10_DINT” and “Target_01_REAL ~ Target_10_REAL” - will be controlled by the Win-GRAF PAC. They represent these variables belong to the 10 Targets .

Data Binding

See Chapter 7 for more details about the Data Binding. If the user wants to open variables of this Win-GRAF PAC for other PACs to read data, these variables that described as above should be dragged into the "Punlic" area of the "Binding" window, and then assign a ID for them. To be controlled correctly by the Schedule-Control configurations, the "Identifier" number **Must be** set from 5001 to 5030. After that, the system will deal with the schedule controls for ten targets according to the settings in the Schedule-Control Utility.

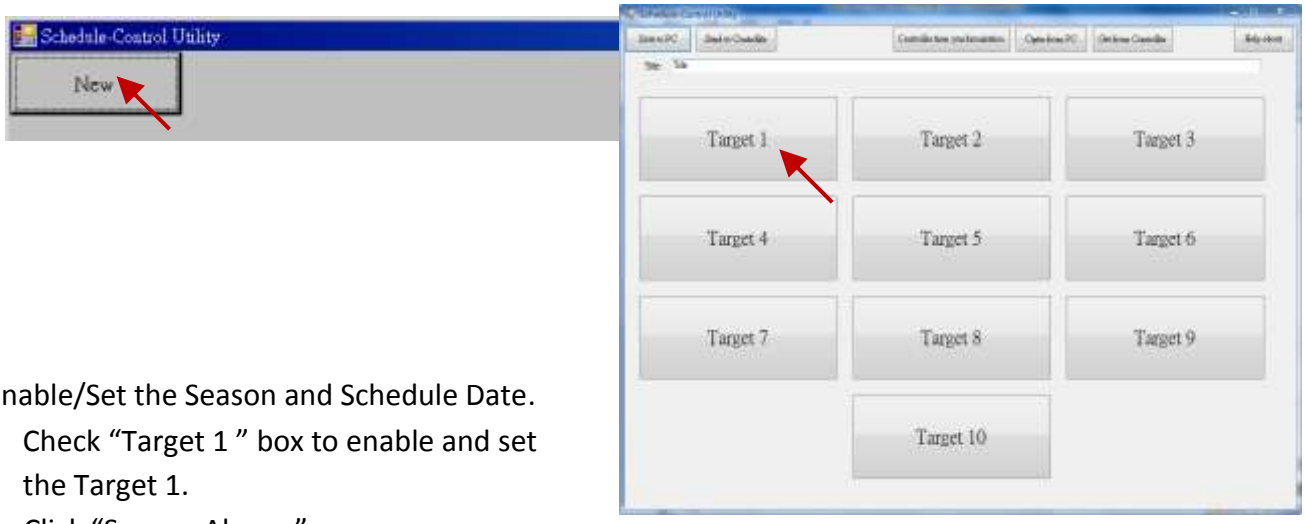


Note: **If the user want to publish Win-GRAF variables to allow the eLogger HMI to get data**. See Section 3.1 To Enable the Win-GRAF PAC as a Modbus TCP Slave for more details.

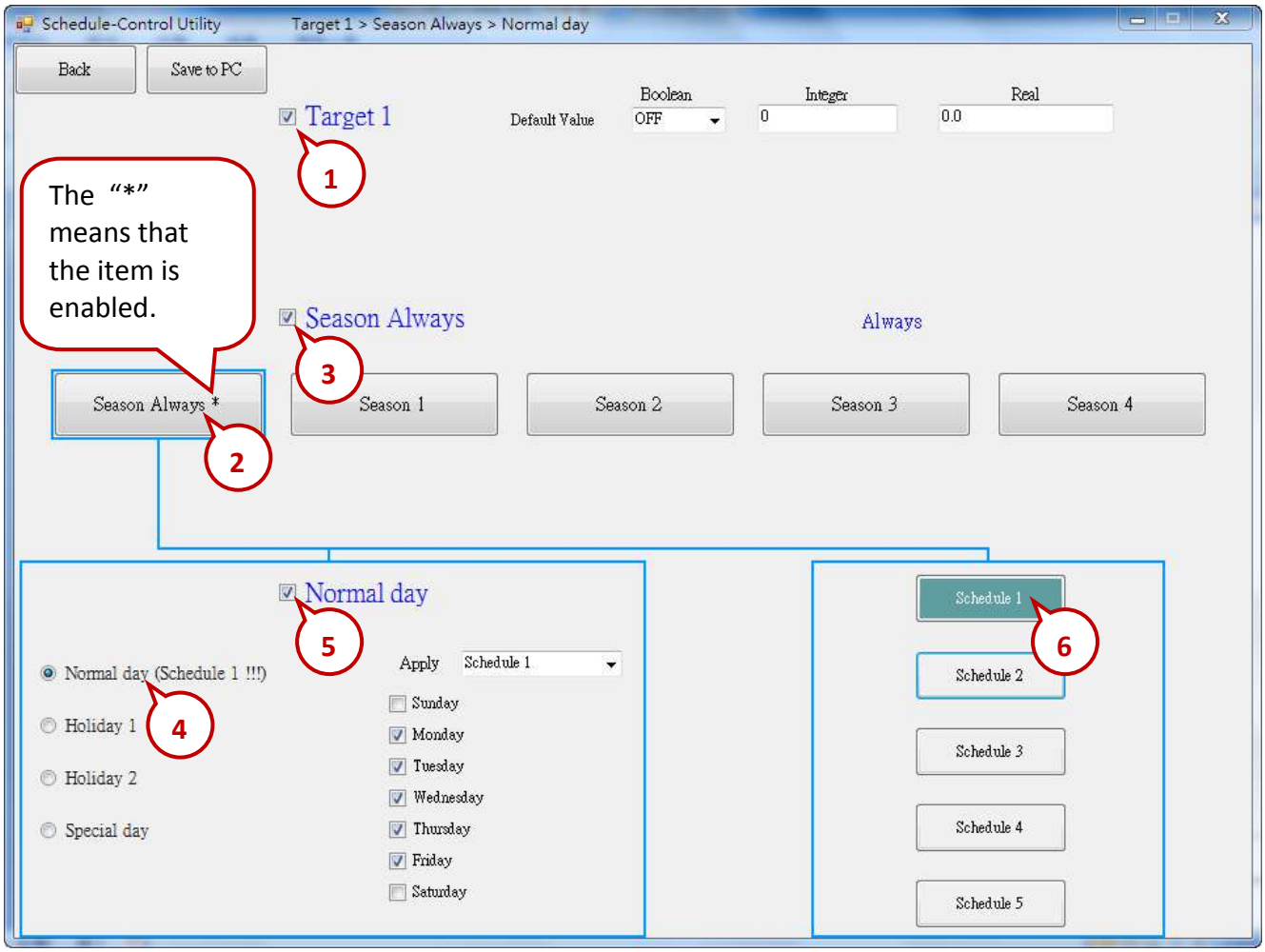
17.3 Edit Schedule Configurations by the Schedule-Control Utility

Here shows a simple example to use the Schedule-Control Utility, refer the Section 17.5 for details.

1. Execute the Schedule-Control Utility (Schedule_in_PC.exe) in the PC.
   (Click "New" and click "Target 1" to open the Schedule setting window for the "Target 1").



2. Enable/Set the Season and Schedule Date.
   (1) Check "Target 1 " box to enable and set the Target 1.
   (2) Click "Season Always".
   (3) Check "Season Always" box to enable it.
   (4) Click "Normal Day" item (Normal day is usually used for Monday ~ Friday.)
   (5) Check "Normal Day" box to enable it and then set proper settings (e.g. Monday ~ Friday).
   (6) Click "Schedule 1" to set the schedule period for the "Schedule 1".



3. Set the Schedule Period

After selecting "Schedule 1" in the step2 – (6), do the following steps.

(1) Check “01” to enable the setting for the No. 01 Time Period of the Schedule 1.

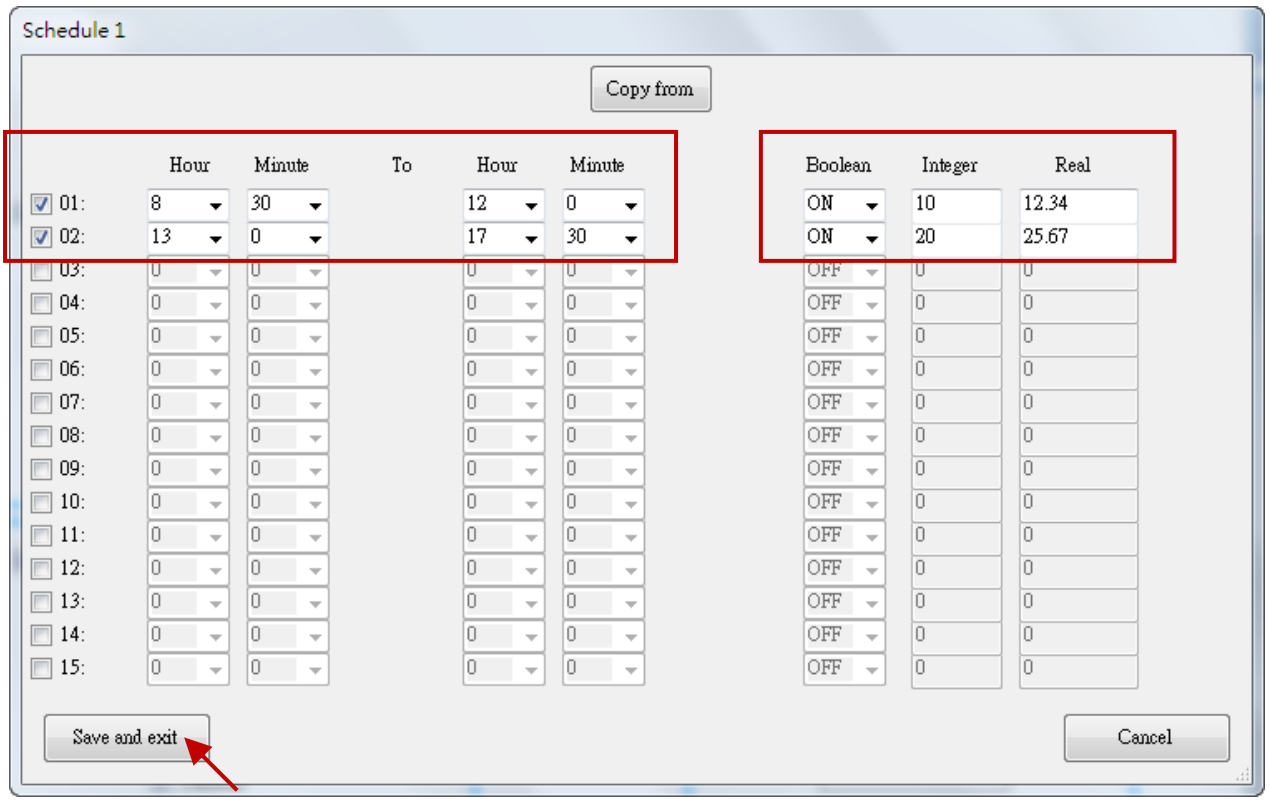
(A) Set up the time as the figure below, or the time which easily for testing.

(B) Set up the Boolean, Integer, Real variables to the values that you want to control, or follow the setting in the figure below.

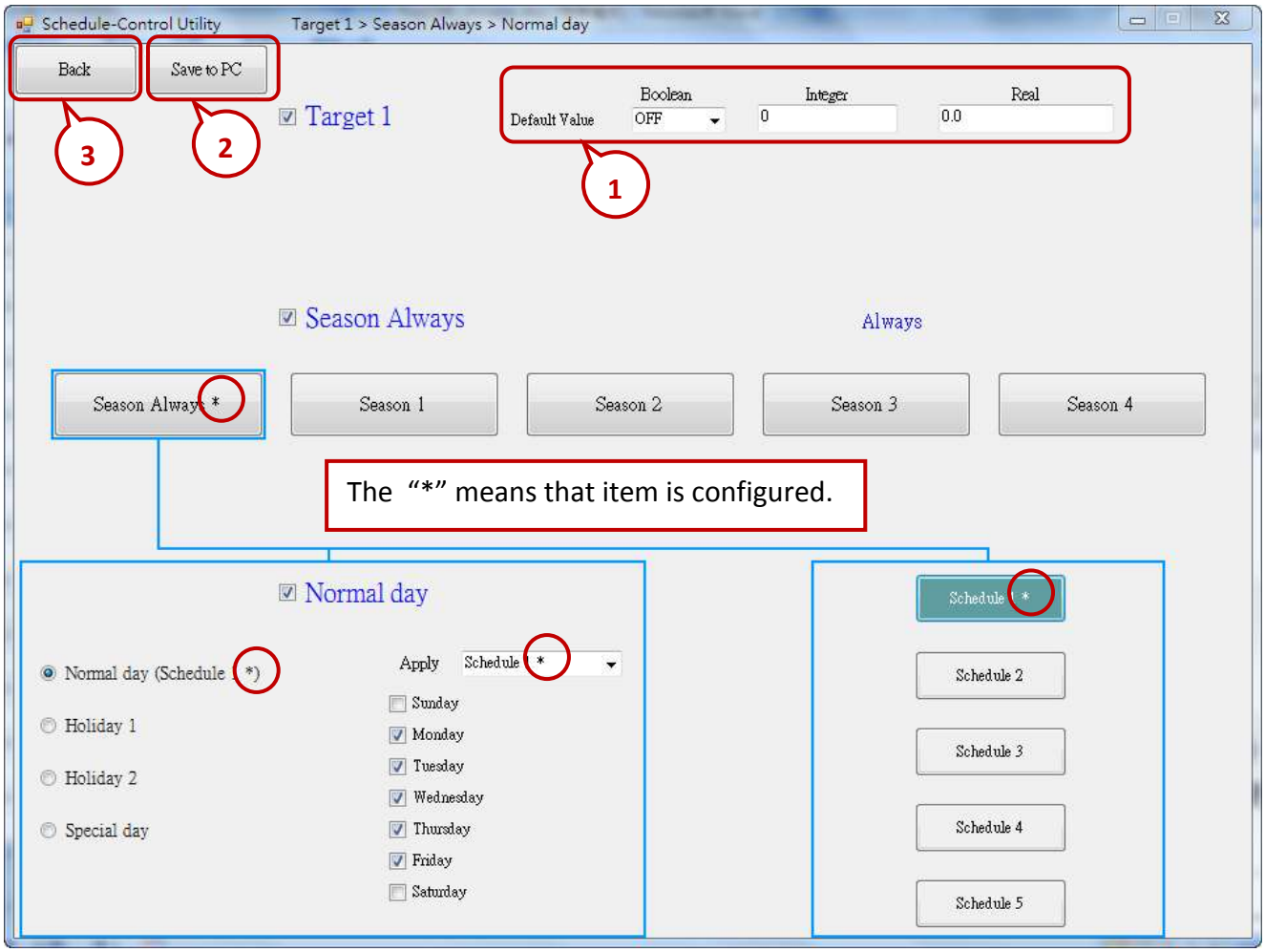
(2) Check “02” to enable the setting for the No. 02 Time Period, such as the step (1).

Each schedule can set up a max. of 15 Time Periods.

After completing the settings, click “Save and exit” to save and exit this window.



4. Then, it will go back to the previous setting window as the figure below. And, the “*” symbol show on the screen means the season or schedule has been configured.



5. "Default Value" (in the upper right) is for the default setting. If the current date is not found in any Schedule setting or the date is found, however, its time period is not found in the related Schedule 1 ~ 5, the Target device will be controlled follow the "Default Value". The "Default Value" in this demo project is "Boolean: OFF, Integer: 0, Real: 0.0" .

Advantage of the Default Value:

Utilizing the "Default Value" can reduce the amount of the Periods setting in the Schedule 1 ~ 5.

Ex: The following example sets 5 Periods in the Schedule 1.

(1) 00:00 ~ 08:00 OFF 0 0.0
(2) 08:00 ~ 09:50 ON 0 0.0
(3) 09:50 ~ 10:00 OFF 0 0.0
(4) 10:00 ~ 11:50 ON 0 0.0
(5) 11:50 ~ 24:00 OFF 0 0.0

If utilize the "Default Value" as "OFF, 0, 0.0", the user just needs to set 2 Periods as below.

(1) 08:00 ~ 09:50 ON 0 0.0
(2) 10:00 ~ 11:50 ON 0 0.0

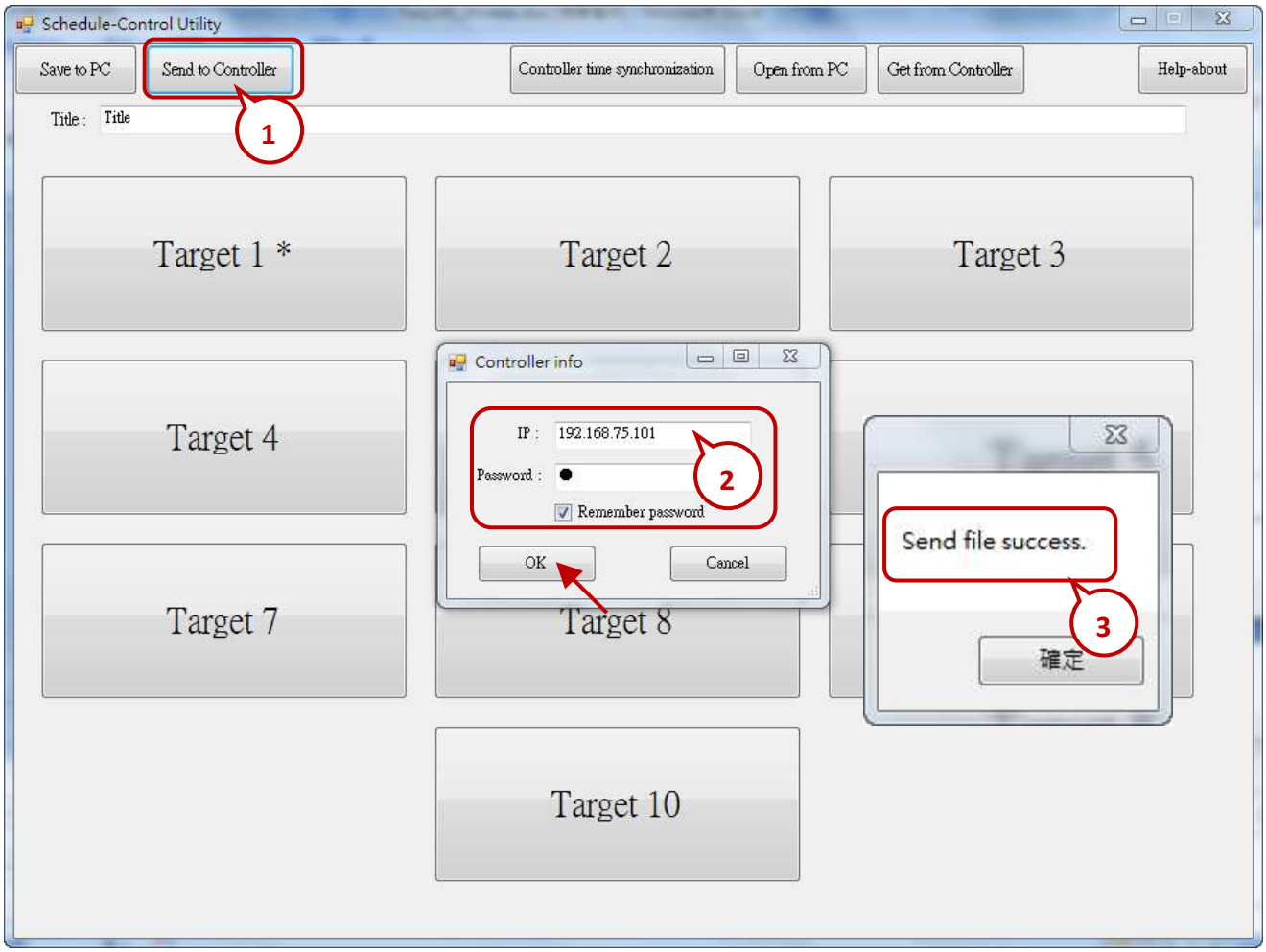
6. After completing all settings, click "Save to PC" to save a configuration file in PC. (This demo uses "test1.txt")
7. Click "Back" to return to set up other Targets. (This demo sets Target 1 only)

17.4 Testing the "demo_schedule" Project

This section shows the way to implement the Win-GRAF project and schedule configuration in the Win-GRAF PAC. Then testing the schedule control.

1. Download the "demo_schedule" project to the PAC by the Win-GRAF workbench.
   (For more information, refer the Section 2.3.5)
2. Download the schedule configuration to the PAC by the Schedule-Control Utility.
   (1) Click "Send to Controller".
   (2) Assign the PAC's IP address (remember to fill in your PAC's IP address)
   Set a password (This demo uses "0")
   Check "Remember Password" can remember this password
   Click "OK" to send the schedule setting to the PAC.
   (3) If success, it will pop up a "Send file success" window.

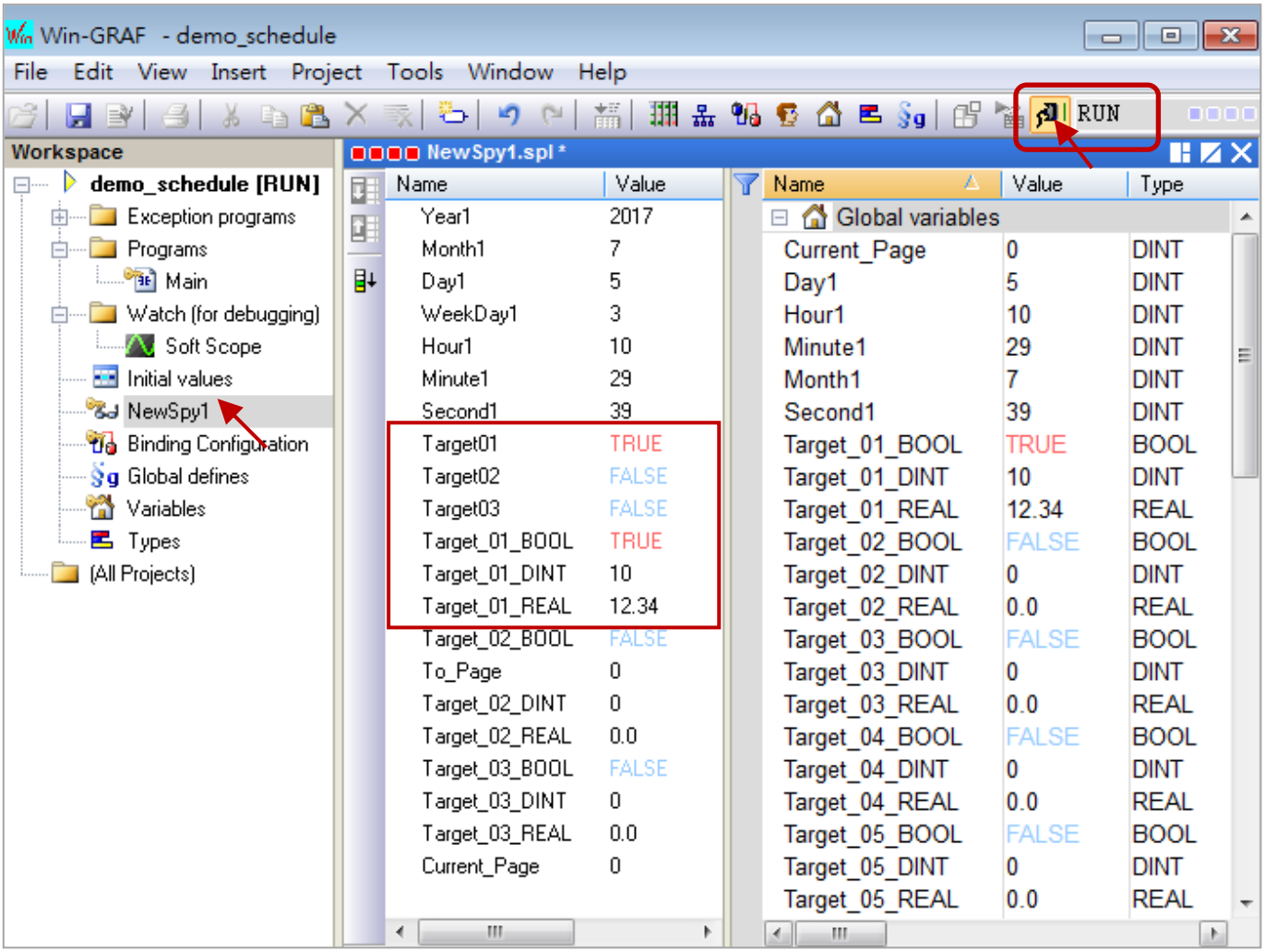
Note: Before downloading the project, make sure the Schesule-Control Utility on PAC has been activated. Moreover, add the \System_Disk\Win-GRAF\Schedule_in_PAC.exe path in the Auto Execution page of the PAC Utility.



3. **Test the Win-GRAF project:**

Click “On Line” to connect the PAC by the Win-GRAF workbench, then open the “NewSpy1” window. If the connection is fine, we can see variables - Target_01_xxx ~ Target_10_xxx are controlled properly by the schedule configurations which is set by the “Schedule-Control Utility”.

The user may use the “Schedule-Control” Utility to modify the schedule configurations and then download to the Win-GRAF PAC to see if those variables are controlled well.



17.5 Configurations of the Schedule-Control Utility

17.5.1 Address for each Target Variables

The Schedule-Control Utility can configure max. 10 Target 's schedule. Each Target contains one BOOL variable, one DINT variable and one REAL variable.

To enable the schedule-control in the Win-GRAF PAC, first add a "Schedule" in the "I/O boards" windows (see Section 17.2). The user can declare all required variables in the "Variables" window, and add these variables in the "Binding" window and then assign correct **"Identifier" number 5001 ~ 5030** (see Section 17.2 - Variable declaration). After downloading the Win-GRAF project to the PAC, the scheduling will control these variables well.

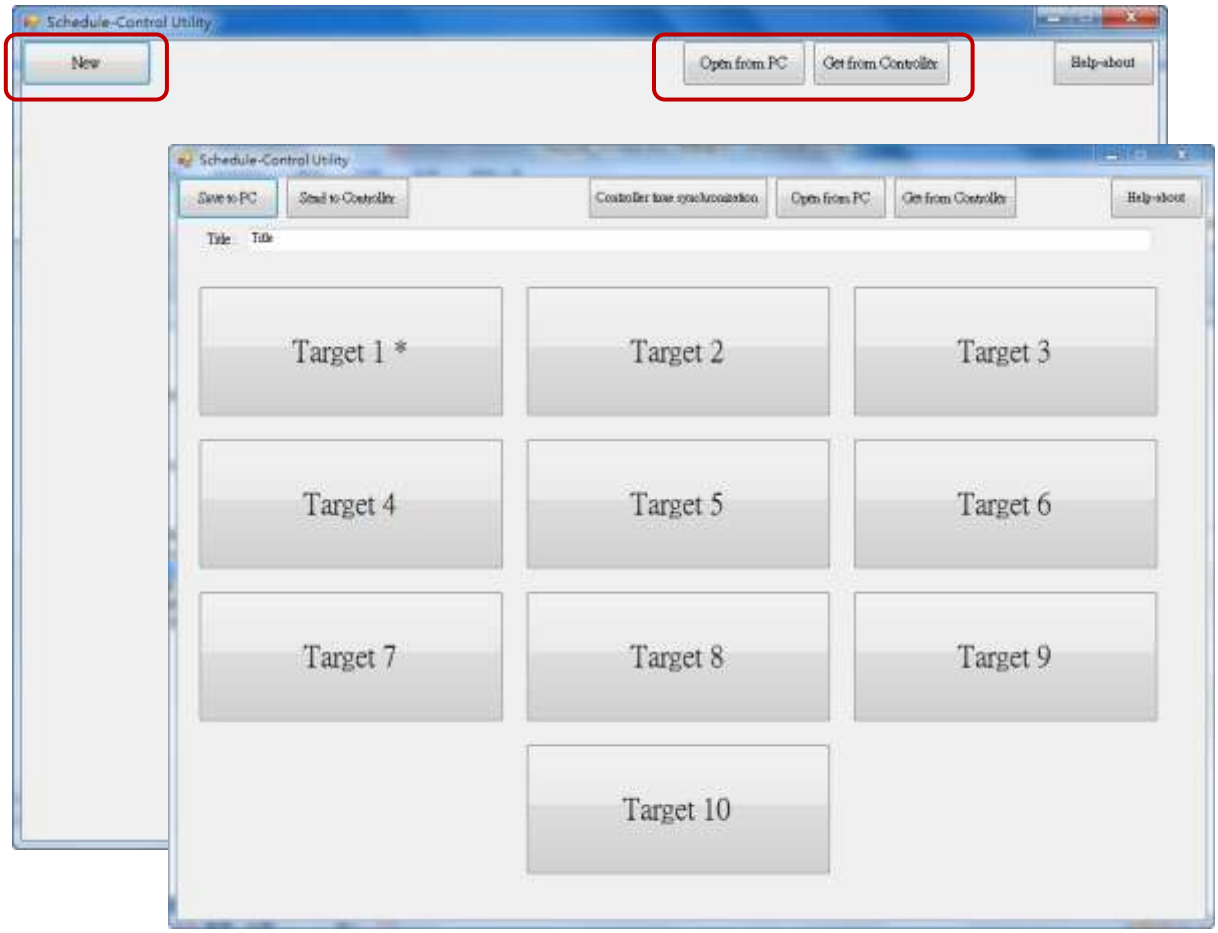
Address	Type	Description	Address	Type	Description
5001	BOOL	BOOL, DINT and REAL variable controlled by **Target 1**	5016	BOOL	BOOL, DINT and REAL variable controlled by **Target 6**
5002	DINT		5017	DINT	
5003	REAL		5018	REAL	
5004	BOOL	BOOL, DINT and REAL variable controlled by **Target 2**	5019	BOOL	BOOL, DINT and REAL variable controlled by **Target 7**
5005	DINT		5020	DINT	
5006	REAL		5021	REAL	
5007	BOOL	BOOL, DINT and REAL variable controlled by **Target 3**	5022	BOOL	BOOL, DINT and REAL variable controlled by **Target 8**
5008	DINT		5023	DINT	
5009	REAL		5024	REAL	
5010	BOOL	BOOL, DINT and REAL variable controlled by **Target 4**	5025	BOOL	BOOL, DINT and REAL variable controlled by **Target 9**
5011	DINT		5026	DINT	
5012	REAL		5027	REAL	
5013	BOOL	BOOL, DINT and REAL variable controlled by **Target 5**	5028	BOOL	BOOL, DINT and REAL variable controlled by **Target 10**
5014	DINT		5029	DINT	
5015	REAL		5030	REAL	

17.5.2 Target Configuration

Every Win-GRAF WinCE PAC can control maximum 10 “Target” (Target 1 to Target 10) devices. First, execute the Schedule-Control Utility and click “New” to create a new configuration file, the Targets will show as 10 buttons (See the figure below). The default Target names are “Target 1” ~ “Target 10”. One Target can set up the schedules to fit different Seasons. The Target button will show a “*” to distinguish it is enabled.

Addition to “New” a configuration file, the user can open an existing file in PC or get from the PAC.

New:	Create a new file.
Open from PC:	Open an exist configuration file from PC.
Get from Controller:	Get an existing configuration file from PAC (required enter the PAC 's IP and password) and then to show on the PC.

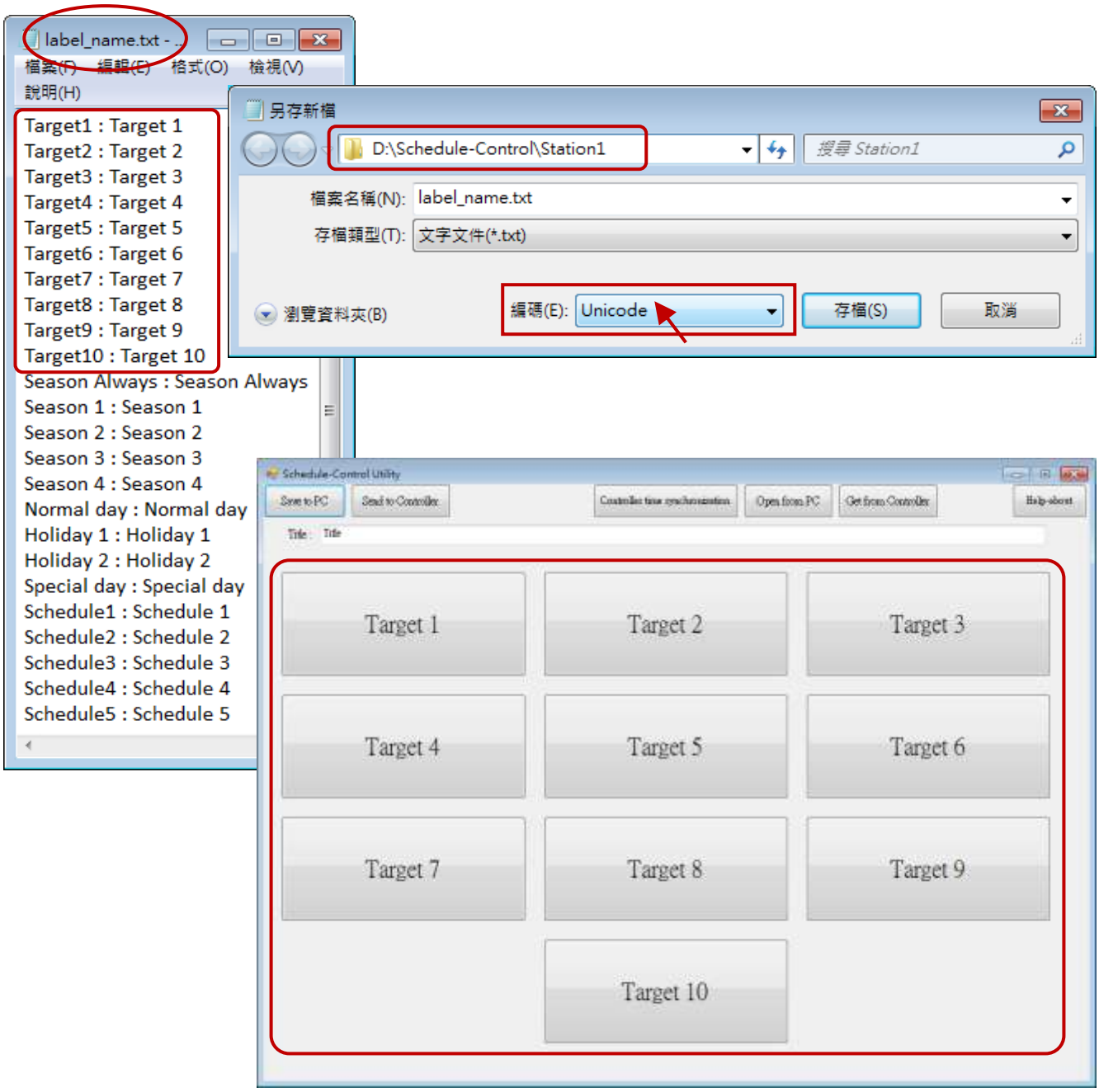


Change the Target Name to meet the needs of the field:

User can change the name of the Target, Season or other items to fit for the equipment at the application field. Please create a text file named “**Label_Name.txt**” (as the figure below) and save it in the same folder with the Schedule-Control Utility “Schedule_in_PC.exe” (e.g., D:\Schedule-Control\Station1\Label_Name.txt).

Notes for creating the file "Label_Name.txt":

1. If this file does not exist, the Target shows the default name (e.g., Target 1, Target 2).
2. In this file, change the target name (e.g., " Factory ", the prefix/suffix of spaces will be erased.) or the name of other items (e.g., Season, Normal day, Holiday, Schedule, etc.) after the colon (" : ").
3. The user can create and edit it by using MS Notepad or other editors, but must select the "Unicode" format when saving it.
4. On the PAC, copy this file into the same folder with the Schedule_in_**PAC**.exe, i.e., \System_Disk\ Win-GRAF\.

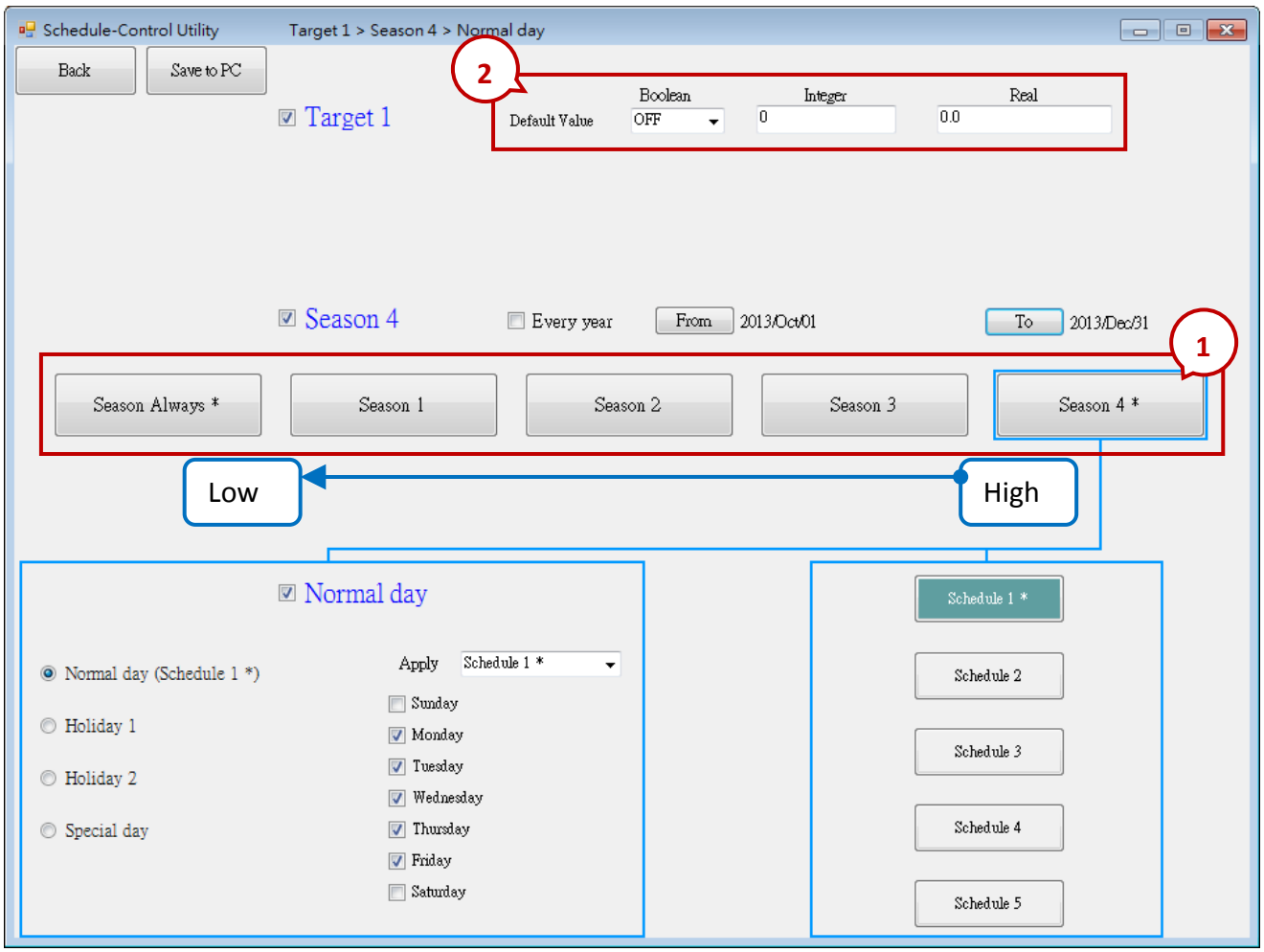


17.5.3 Season Configuration

Each “Target” (1 ~ 10) includes the “Season Always”, “Season 1”, “Season 2”, “Season 3” and “Season 4” setting items. It is recommend to check "Season Always" that means to enable the year-round schedule.

The Searching Priority of Seasons:

1. PAC will first search the **Season 4** (if it is enabled)
   If found the current date in the Season 4, then do the Boolean/Integer/Real control.
2. If not found, then search the **Season 3, Season 2**..., at last search the **Season 1**.
3. If not found, then search the **Season Always** to do its control.
4. If not found the current date in this Target, then do the “Default Value” control.

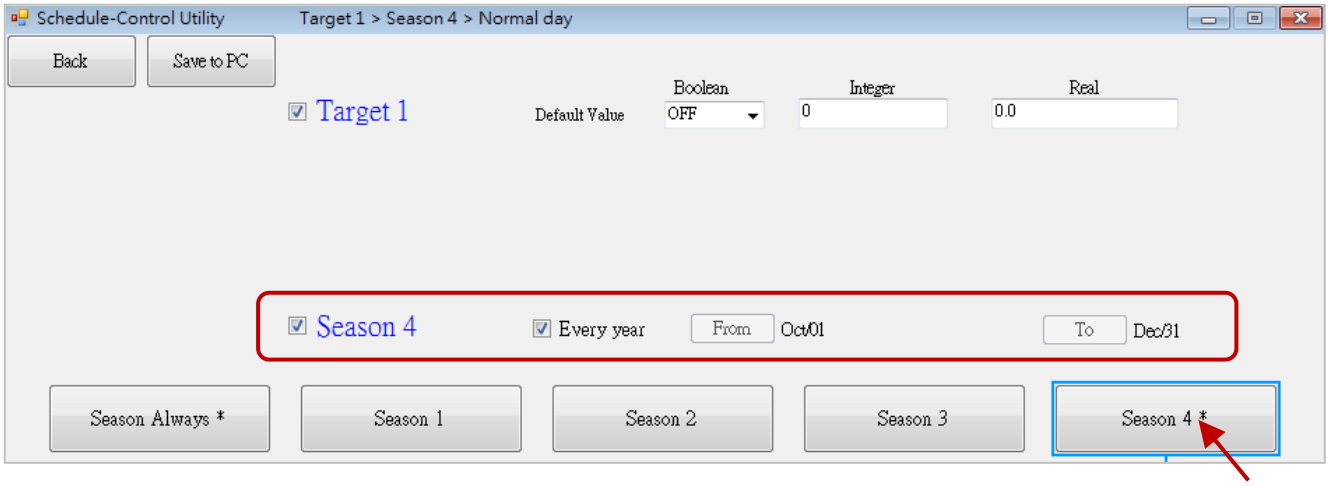


Season Setting:

Season 1 ~ 4 need to set its “Date Period”. After completing the settings, recommand to check the “Every Year” option to apply the Date Period every years.

Note 1: The Date Periods of the 4 Seasons must not overlap.

Note 2: If the "Every Year" is checked, the system diagnoses the overlap of Month/Day only, not the year. If the "Every Year" is not checked, it will diagnose the “From” Year/Month/Day should be earlier than the “To” Year/Month/Day.



Note: Unchecked the "Every Year" can modify the date periods, and take notice of the date order.

For example:

1. **The Correct Setting:**
   User can check "Every Year", so that the setting will be used for every year.

Season 1	01/01 ~ 03/31
Season 2	04/01 ~ 07/15
Season 3	07/16 ~ 09/30
Season 4	10/01 ~ 12/31

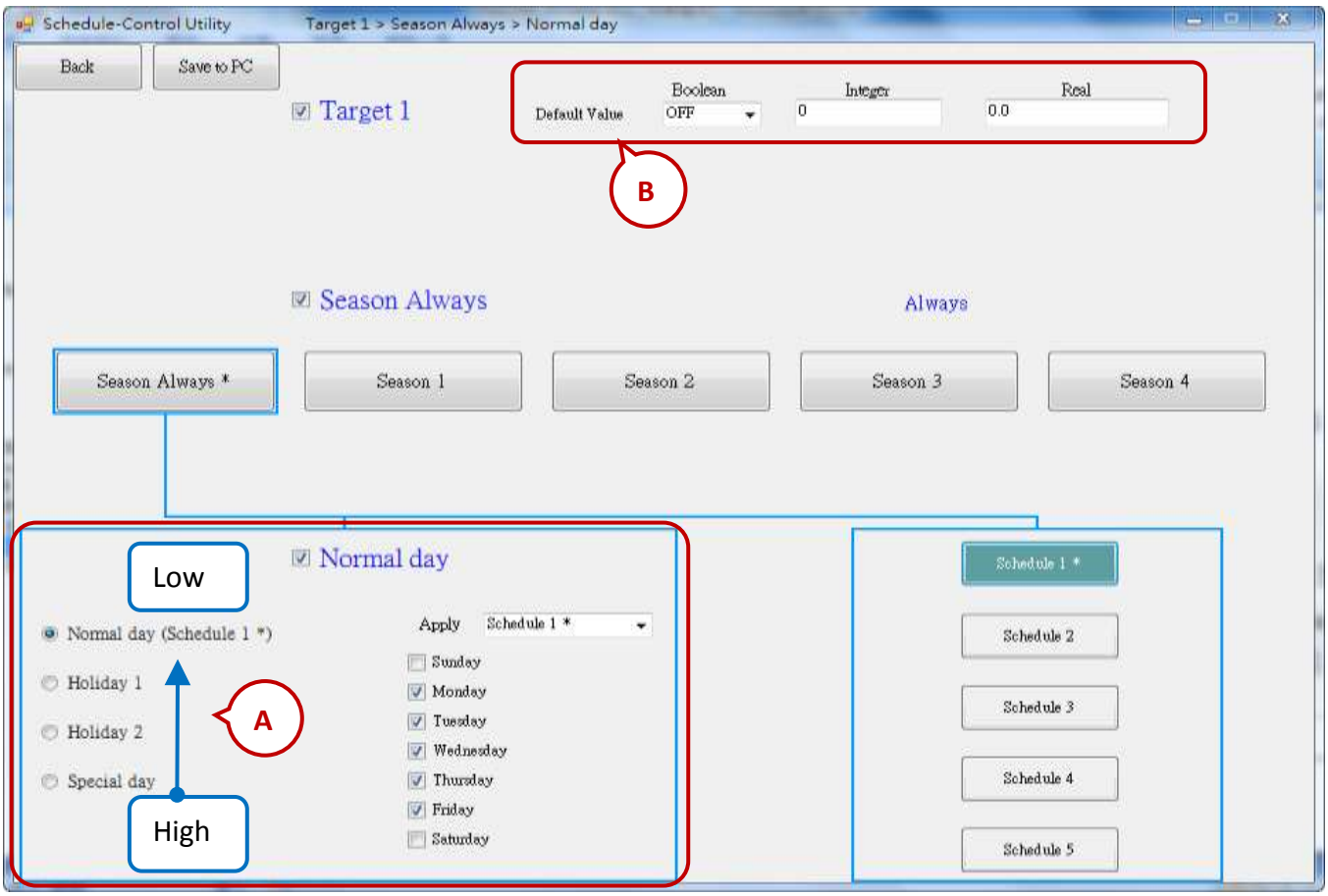
2. **The Wrong Setting:**
   Because the Season 2 overlaps the date of Season 1 from 03/16 to 03/31.

Season 1	01/01 ~ **03/31**
Season 2	**03/16 ~** 07/15
Season 3	07/16 ~ 12/31
Season 4	Disabled

17.5.4 Normal Day / Holiday / Special Day Configuration

There are Normal day, Holiday 1, Holiday 2 and Special day in each Season. **When enabling the setting, users must choose a Schedule number (1 to 5) to apply the time settings.**

Normal day	The normal days are Monday to Friday.
Holiday 1	Normally set to Saturday and Sunday.
Holiday 2	In some workplace, there are different holidays, e.g., Wednesday.
Special Day	Set the schedule for local holidays or the adjusted working-day. E.g., Oct. 10, Jul. 4, Oct. 1, Dec. 25, etc. A maximum of 50 days can be set per Season.



A. The Searching Priority of Normal Day / Holiday / Special Day:

The PAC will first search **"Special day"**. If the date is not found in this Special day setting, then search **"Holiday 2"**, then **"Holiday 1"**, and then **"Normal day"**.

B. Default Value for Boolean / Integer / Real:

Each Target must set the default value for the Boolean, Integer and Real variables. These default values will be applied when the PAC cannot find any available "Date Period" or "Time Period" in the enabled "Season" setting. Then the PAC follows the setting of the Default Value. Usually, the Default Value of Boolean is set to be "OFF", the Integer and Float value are set to be "0". User can set the different Default Value by the case.

C. Date Setting for Normal day / Holiday 1 / Holiday 2:

Note that NO OVERLAP. For example,

The Correct Setting:

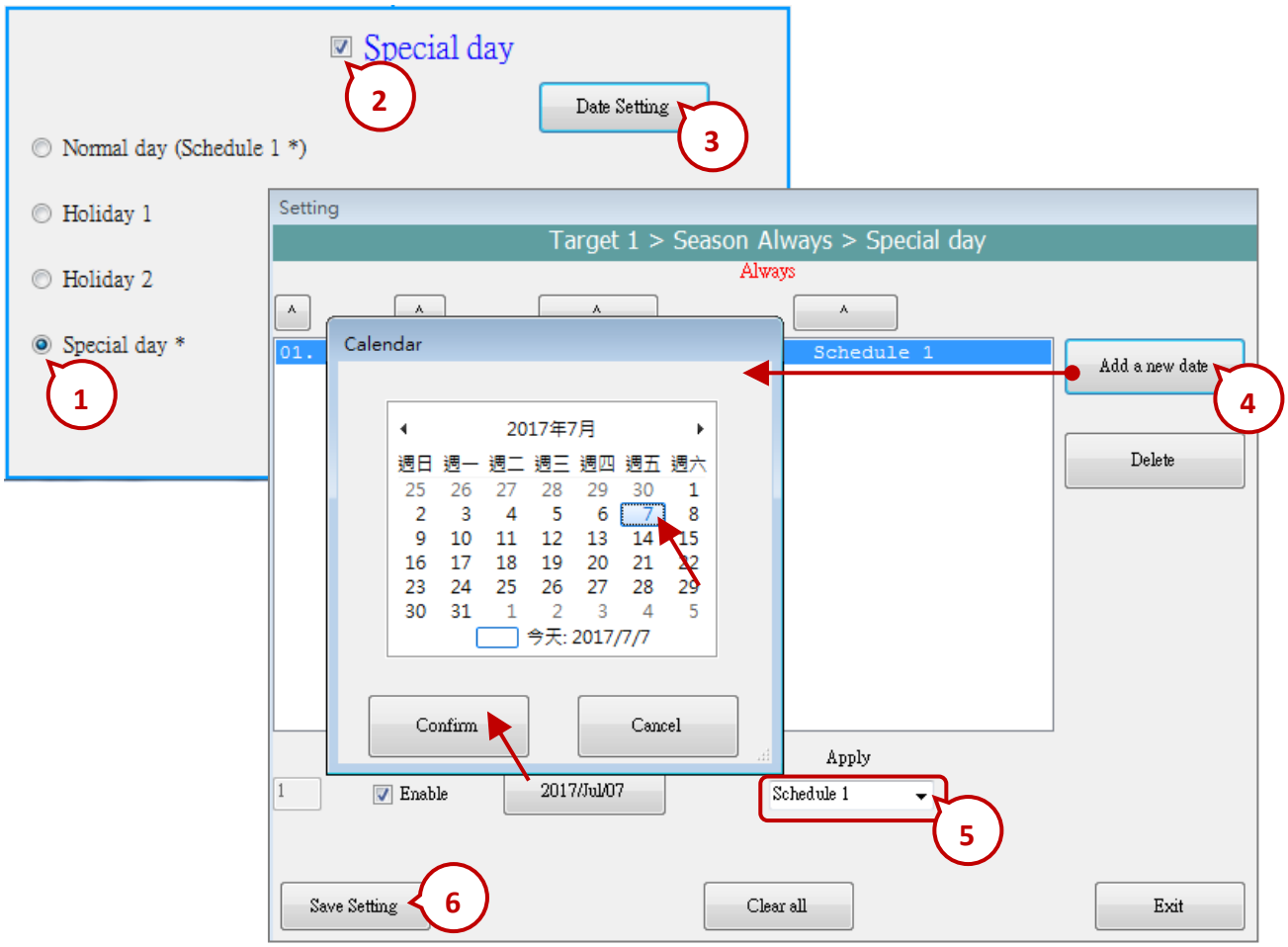
Normal day	Monday, Tuesday, Wednesday, Thursday, Friday
Holiday 1	Sunday, Saturday
Holiday 2	Disabled

The Wrong Setting: (Because **"Friday"** overlaps in the setting of "Normal day" and "Holiday 2".)

Normal day	Monday, Tuesday, Wednesday, Thursday, **Friday**
Holiday 1	Saturday
Holiday 2	Sunday, **Friday**

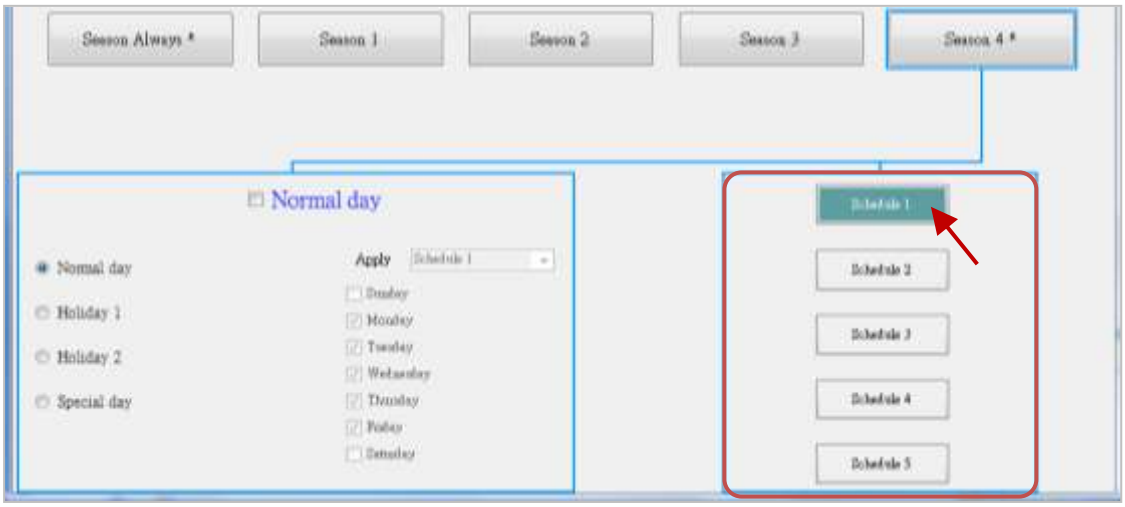
D. Date Setting for Special Day:

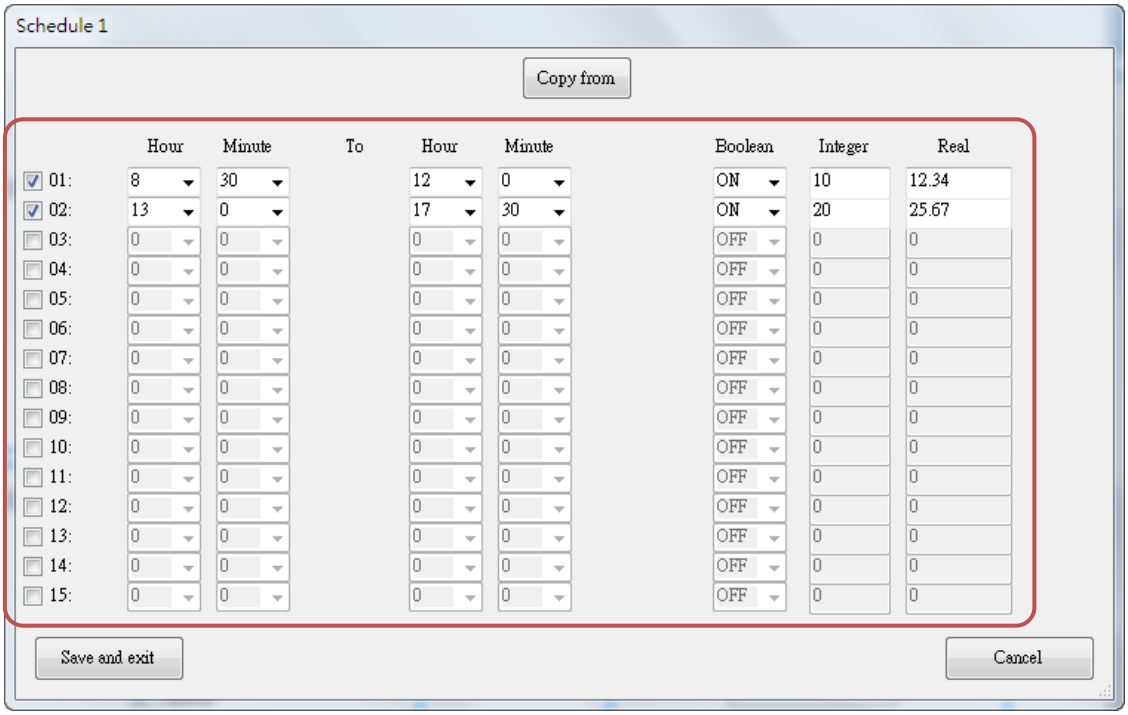
The "Special day" is for special schedule, such as the special holidays or make-up workdays. Each Season can set maximum 50 Special days. The searching priority of the "Special day" is higher than the priority of Holiday 2 and Holiday 1 and Normal day. Each enabled "Special day" date must select a Schedule number (1 ~ 5) to be applied.



17.5.5 Schedule Configuration

Each Season can set up maximum 5 Schedules (Schedule 1 ~ 5), and each Schedule can set up maximum 15 Time Periods. The time unit is “minute”, in the range of “00:00 ~ 24:00”.





EX: The following setting is correct.

No.	Time Period	Boolean	Integer	Real
01	00:00 ~ **08:00**	OFF	100	30
02	**08:00** ~ 12:00	ON	150	25.5
03	**12:00** ~ 13:00	OFF	120	27
04	**13:00** ~ 17:00	ON	150	25.5
05	**17:00** ~ 24:00	OFF	100	30

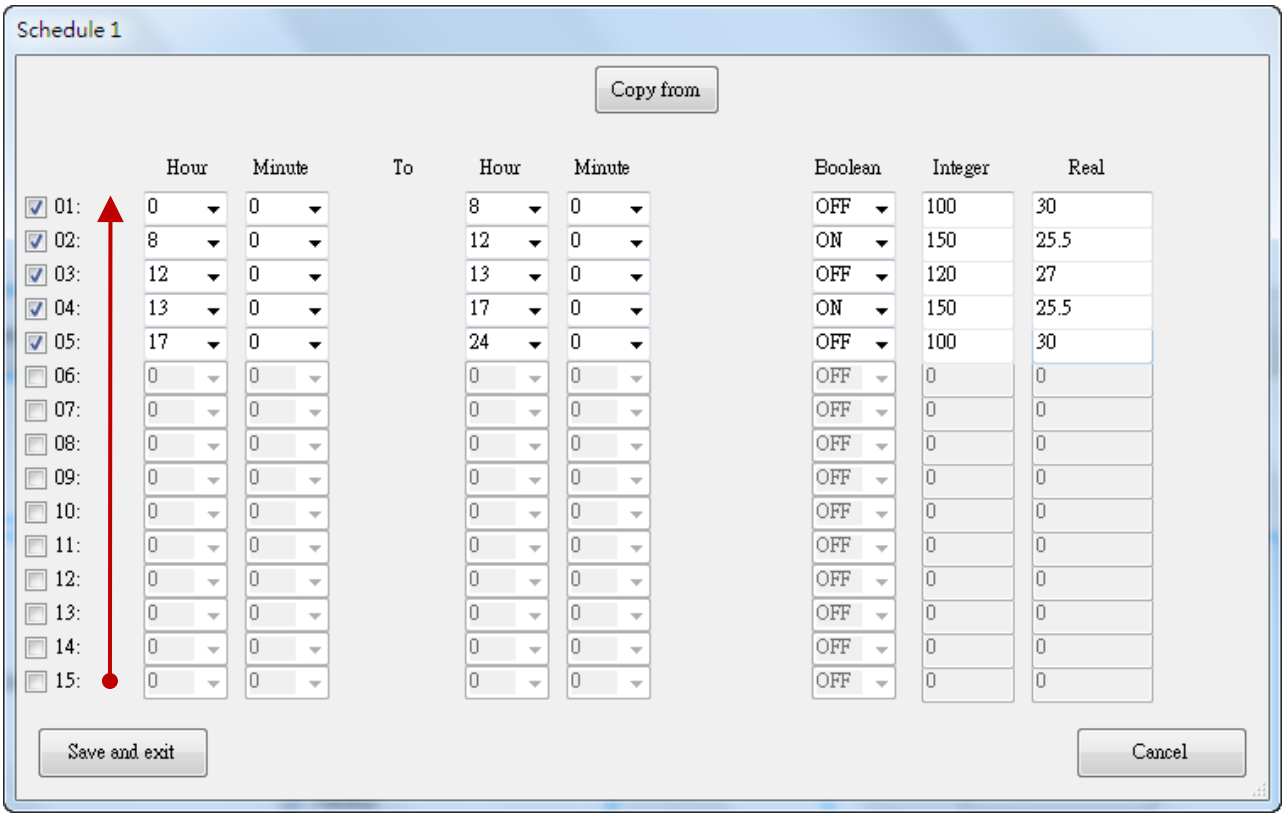
The Searching Priority of Time Period:

The searching priority of the Time Period in the schedule is in the order from the largest number to the smallest number.

For example, the following table shows five Time Periods settings .

No.	Time Period	Boolean	Integer	Real
01	00:00 ~ 08:00	OFF	100	30
02	08:00 ~ 12:00	ON	150	25.5
03	12:00 ~ 13:00	OFF	120	27
04	13:00 ~ 17:00	ON	150	25.5
05	17:00 ~ 24:00	OFF	100	30

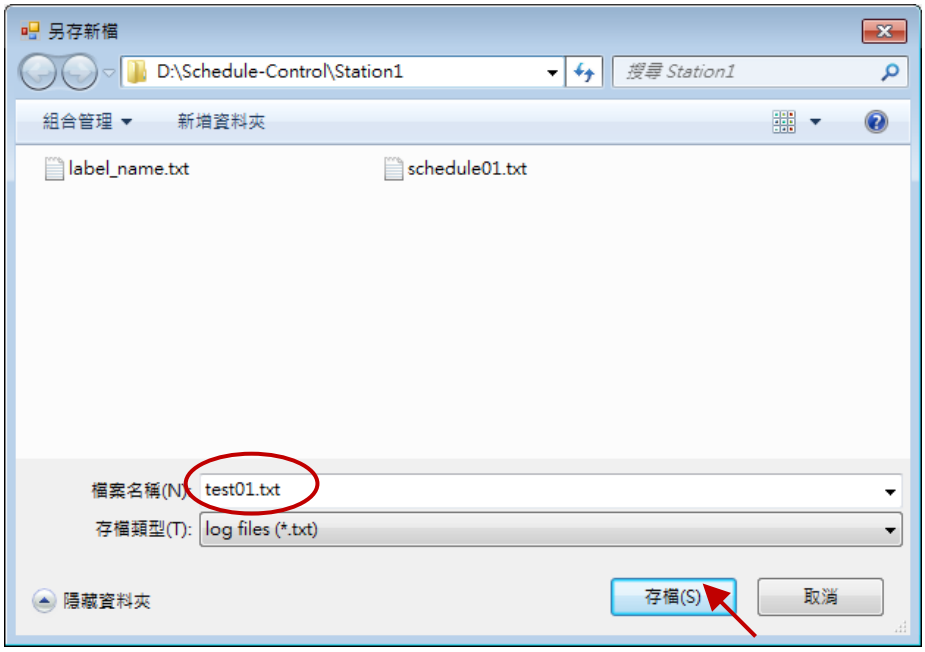
1. The searching will in the order from No. 5 to No. 1 (05 , 04 , 03 , 02 , 01). If the Time Period overlaps, the PAC will follow the larger number setting to control the schedule.
2. If the PAC cannot find the current time in any Time Period in the “15” ~ “01”, it follows the setting of "Default Value".



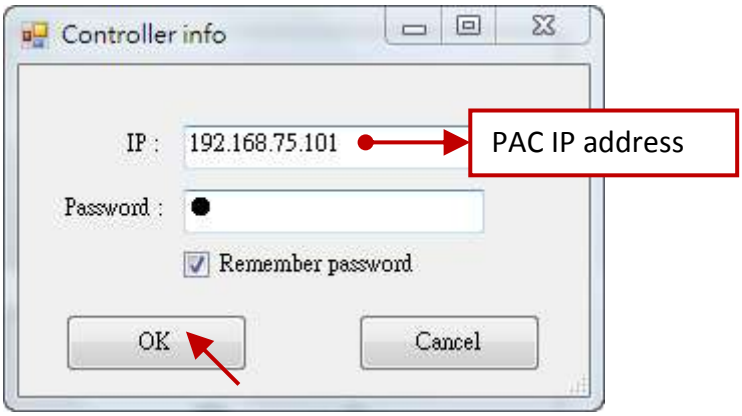
17.5.6 Save and Send the File to the PAC

After completing the configurations, please save and then send it to the PAC:

1. Click “Save to PC” to save the configuration file (“ *.txt ”).

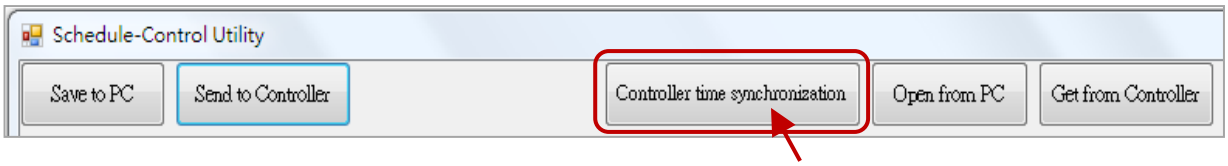


2. Click “Send to Controller” to send the configuration file to the linked PAC. Please assign the PAC IP address and set up the password (default: 0). Check the “Remember password” can save the password for speeding the next sending process.



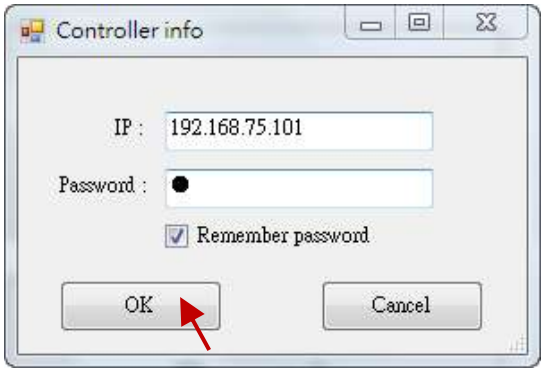
17.5.7 Time Synchronization

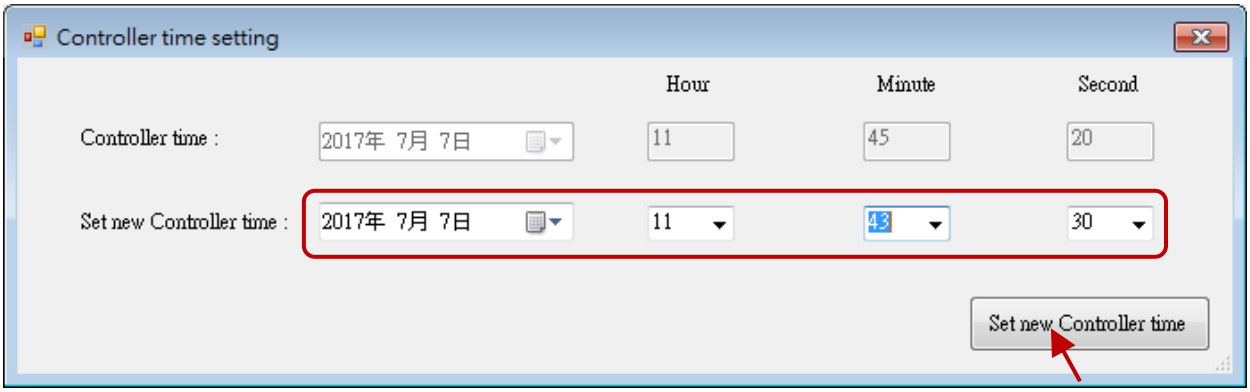
If the PAC has not synchronized the system time after working a long period (e.g. one year), the time may be differ over 10 seconds to a few minutes. For the time synchronization of the controller, the Schedule-Control Utility provides a function to set the PAC time from the PC.



Steps:

1. Your PC must connect to an Ethernet Switch and then to the Win-GRAF PAC by using Ethernet cables.
2. Click “Controller Time Synchronization” button, and enter the current IP address of the PAC and the password (defaults: 0).
3. Set a new date, hour, minute and second.
4. Click “Set new Controller time” button to set the new time to the PAC.

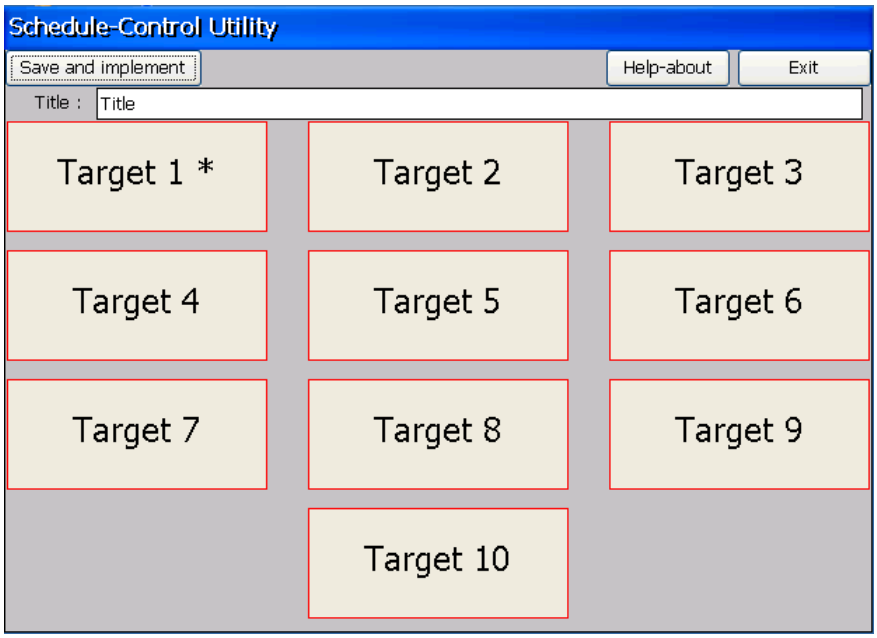




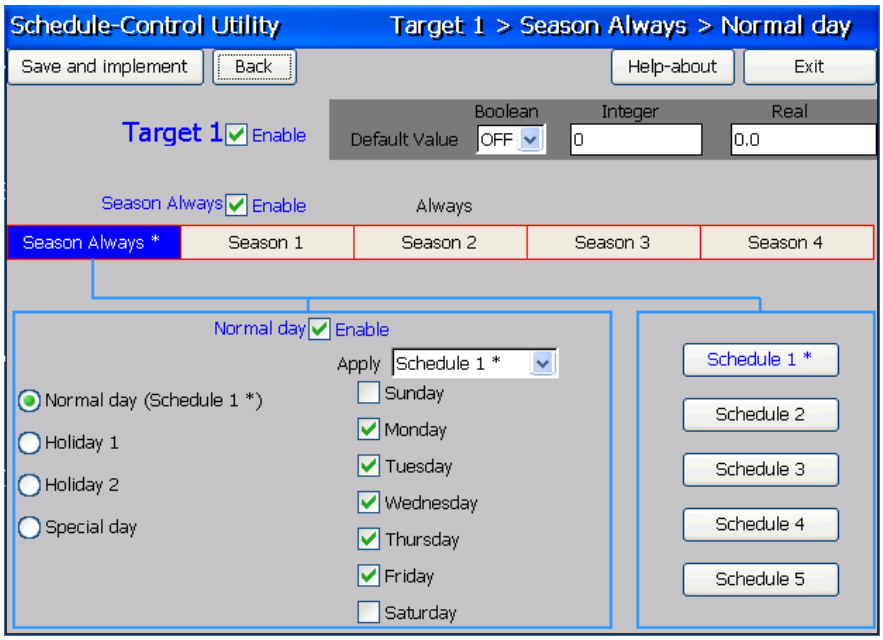
17.5.8 Schedule-Control Utility in PAC Site

The configuration process on the PAC side is similar as the process in the PC side, just a little bit different on the screen. Please refer the Section 17.5.2 ~ 17.5.6.

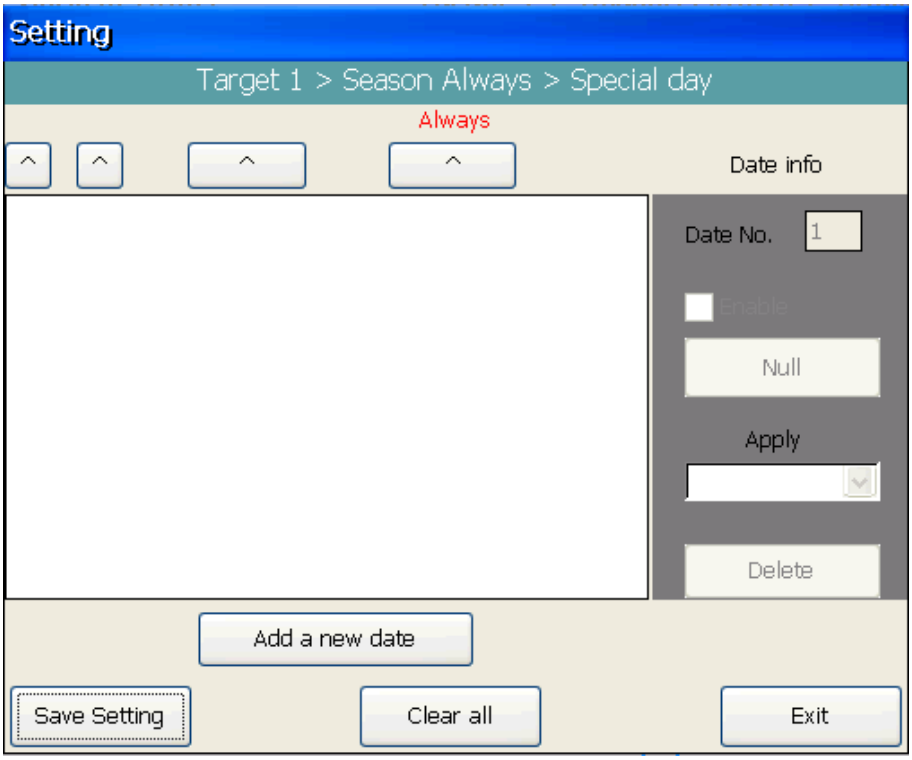
1. Target Configuration:



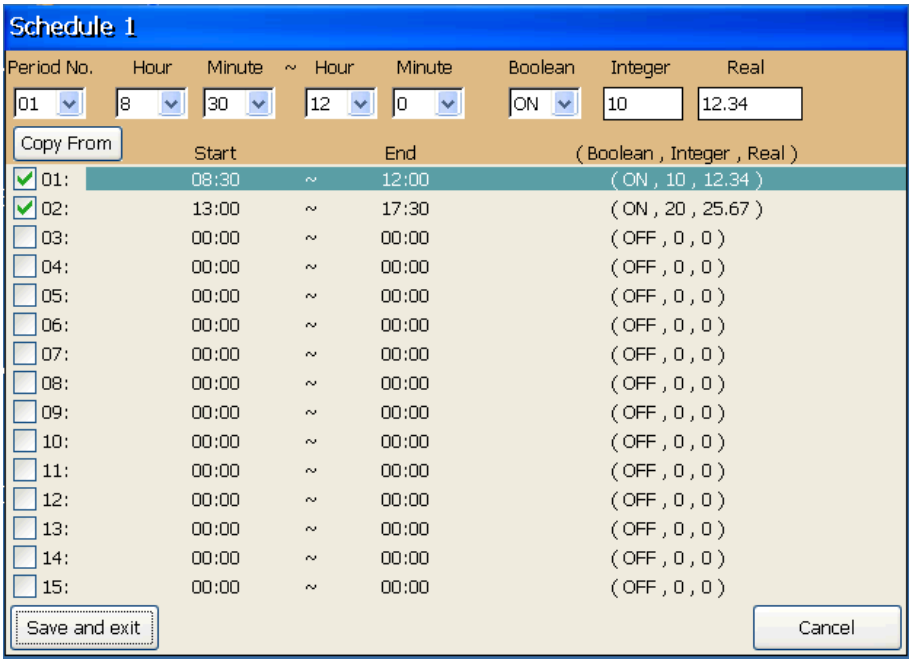
2. Season Configuration:



3. Special Day Configuration:



4. Schedule Configuration:



17.5.9 Using Schedule-Control in the eLogger HMI

eLogger is a free charge and an easy-to-use HMI software platform developed by ICP DAS. It can be used to design the Local HMI and the Web Server HMI for remotely controlling the PAC through a web browser on your PC or cell phone. All Win-GRAF PAC support eLogger HMI.

For instructions on eLogger HMI, visit the following web page for the Win-GRAF FAQ-018 and FAQ-019:

www.icpdas.com > Support > FAQ > Win-GRAF Soft-Logic PAC > FAQ-018, FAQ-019 or http://www.icpdas.com/root/support/faq/win-graf.php

You can download the sample program ("Demo_faq018_all. zip" or demo_faq019_all. zip) directly on the Win-GRAF FAQ page, or in Win-GRAF-PAC-CD (\napdos\win-graf\demo_ project\), and then operate and test the project according to the contents of the document.

Chapter 18 Develop Your Own Function and Function Block

This section described how to use the Visual Studio 2008 development tool to produce a DLL file of your own Function or Function Block. All the related demo files are included in the PAC's CD-ROM, such as the XP-8xx8-CE6, WP-8xx8, WP-8xx8-CE7, WP-5xx8-CE7, and VP-x2x8-CE7 PAC.

Related demo files:

CD-ROM : \napdos\Win-GRAF\demo-project\user_c_lib\

\demo_user_c : The VS 2008 project folder, can be used to build the "user_c.dll" file for your own Function and Function Block. (Refer the Section 18.4)

..\user_c.dll : The pre-compiled DLL file - "user_c.dll" used for the Function ("bytes_to_long") and the Function Block ("long_to_bytes") in this demo.
\wp_vp\user_c.dll : for WP-8xx8, WP-8xx8-CE7, VP-x2x8-CE7, and WP-5xx8-CE7.
\xpc\user_c.dll : for XP-8xx8-CE6.

\user : The Win-GRAF Lib folder, including library files of the Function ("bytes_to_long") and the Function Block ("long_to_bytes") in this demo. (Refer the Section 18.3)

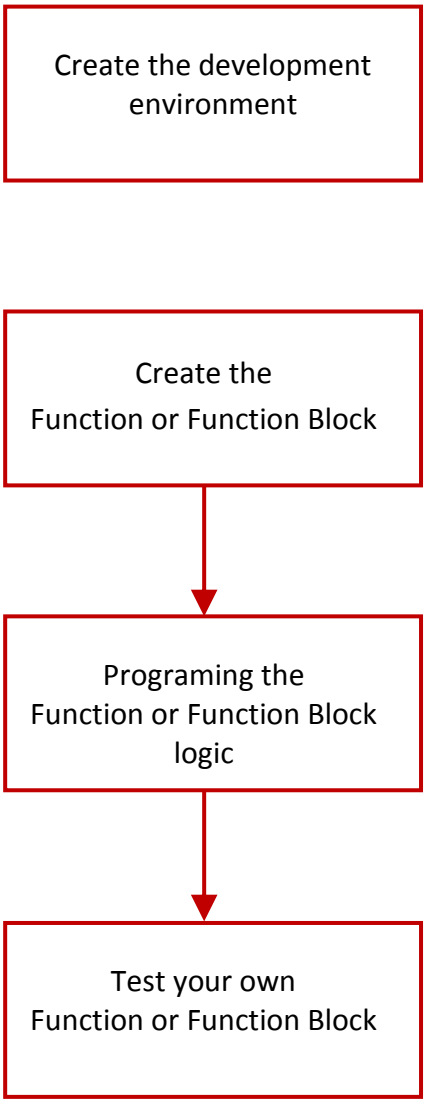
Win-GRAF demo file:

CD-ROM : \napdos\Win-GRAF\demo-project**demo_user_c.zip**

The standard shipments of ICP DAS Win-GRAF PAC do not include the "user_c.dll" file. If you want to add your own Function and Function Block for working with the Win-GRAF logic, you must build a DLL file and copy it to the same folder as the Win-GRAF driver on the PAC (i.e., \System_disk\Win-GRAF\).

After rebooting the PAC, the Win-GRAF driver will load that DLL file to support your Function and Function Block.

18.1 The Development Process of Your Own Function or Function Block



Use VS 2008 and install the correct SDK.
(Refer the Section 18.2)

Use Win-GRAF Library Manager to generate the source code.
(Refer the Section 18.3)

Import the source code generated in the previous step to the VS 2008 project to edit the logic.
(Refer the Section 18.4)

Create a Win-GRAF project that includes your own Function or Function Block, and download it to the Win-GRAF PAC.
(Refer the Section 18.5)

18.2 Creating the Compiler Development Environment

Download the SDK (Software Development Kit):

The user can download the related SDK on the website:

1. For XPAC (XP-8xx8-CE6)
   ftp://ftp.icpdas.com/pub/cd/xp-8000-ce6/sdk/platformsdk/
   (pacsdk_ce_x.x.x_vs2008.msi)

2. For ViewPAC (VP-x2x8-CE7) and WinPAC (WP-8xx8, WP-8xx8-CE7, WP-5xx8-CE7)
   ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/sdk/platformsdk/
   (pac270_sdk_yyyymmdd.msi)

18.2.1 Install the SDK of the ViewPAC or the WinPAC

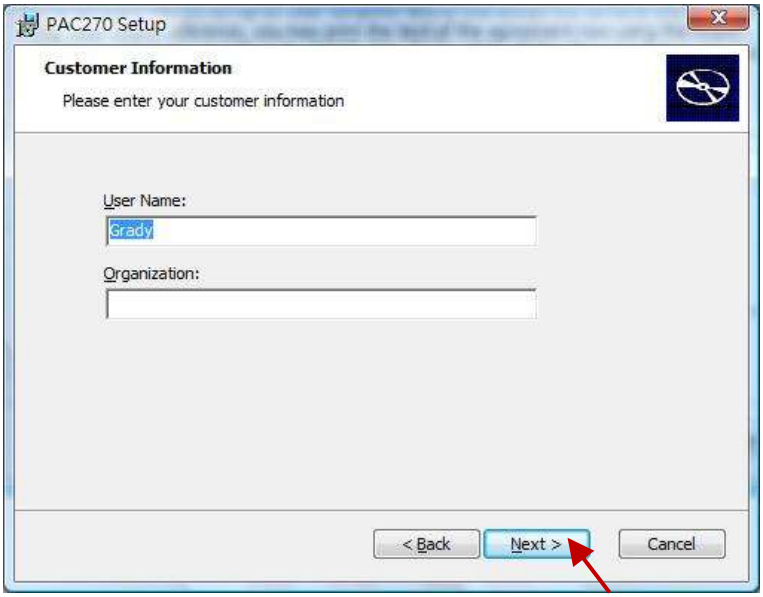
Note: Make sure your PC has been installed the Microsoft VS2008 before doing the following steps.

1. Double click the downloaded SDK file (e.g., pac270_sdk_20121015.msi) to install it to the VS2008.

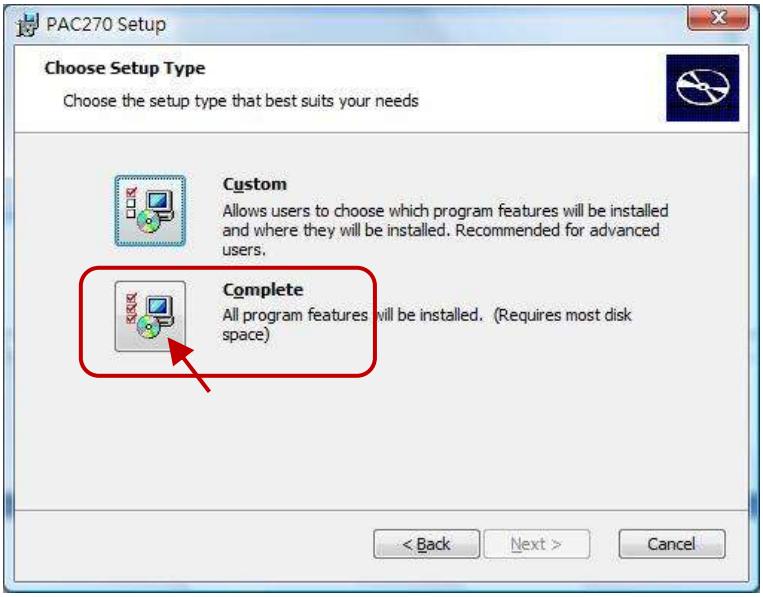
2. Click the "Next" button.

3. Choose the “Accept” radio button and then click the “Next” button.

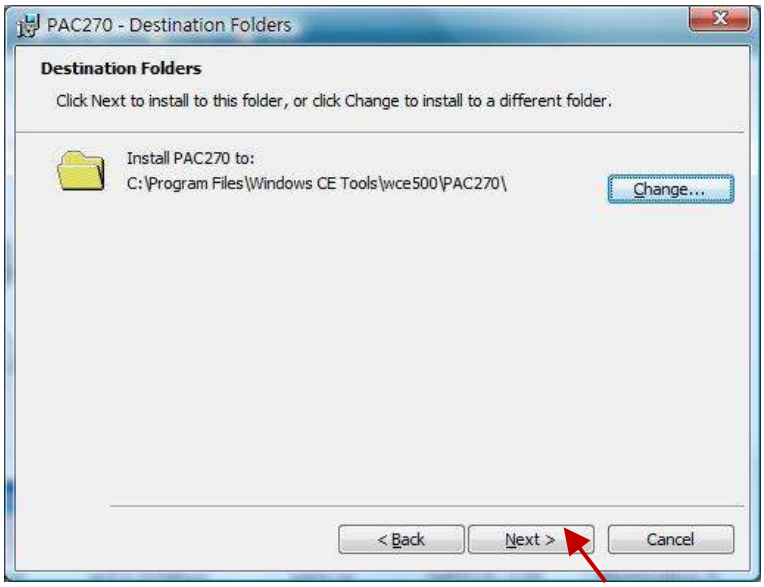
4. Click the “Next” button.



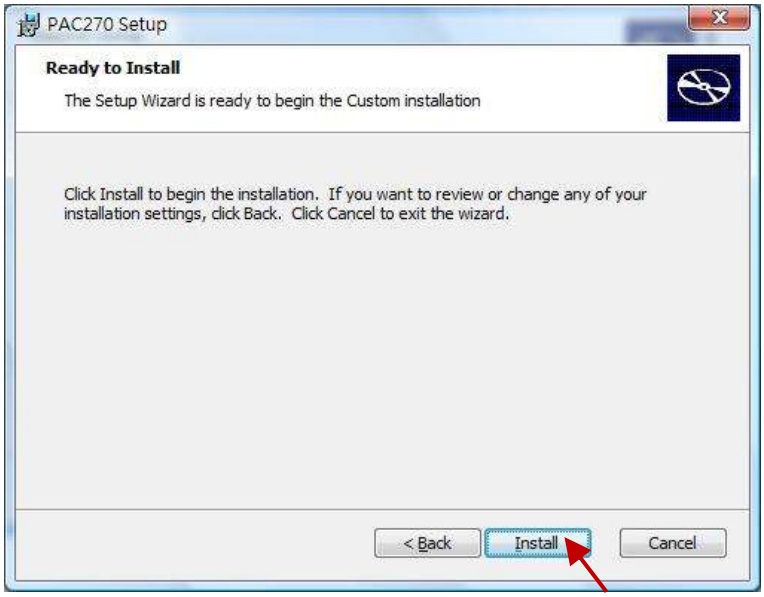
5. Click the “Complete” button.



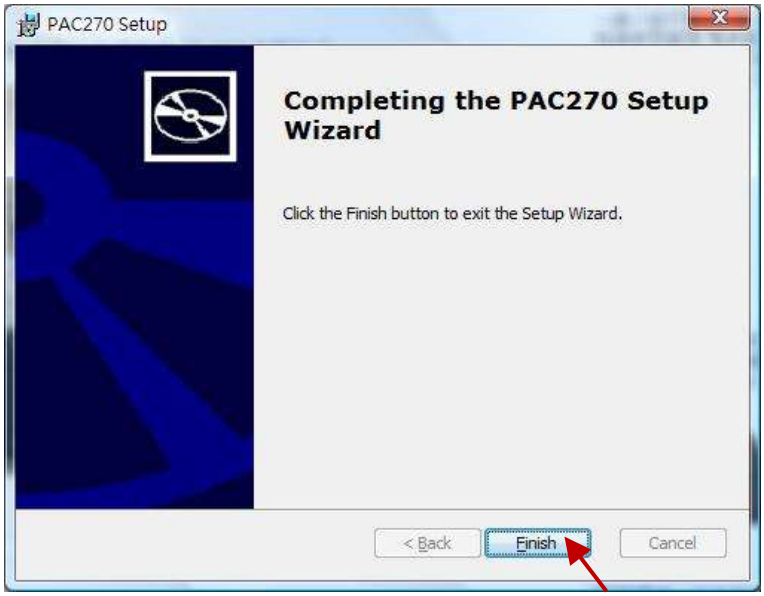
6. Click the “Next” button.



7. Click the “Install” button to install the SDK.



8. After completing the installation, click the “Finish” to quit the procedure.



18.2.2 Install the SDK of the XPAC (XP-8xx7-CE6, XP-8xx7-Atom-CE6)

Note: Make sure your PC has been installed the Microsoft VS2008 before doing the following steps.

1. Double click the downloaded SDK file (e.g., pacsdk_ce_1.4.3_vs2008.msi) to install it to the VS2008.

2. Other steps, refer the Section 18.2.1 – Step (2) ~ (8).

18.3 Define Function or Function Block

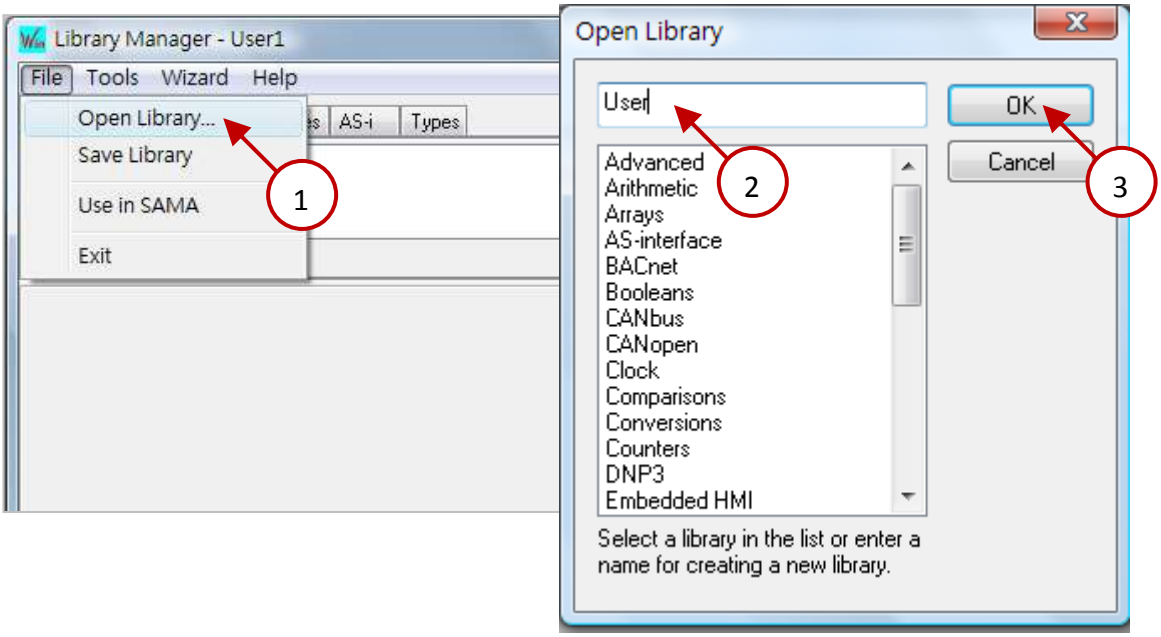
18.3.1 Define Function Lib

This section presents a simple example of creating a library function - “bytes_to_long” that is used to convert four bytes (0 ~ 255) to one long integer (32-bit Signed Integer).

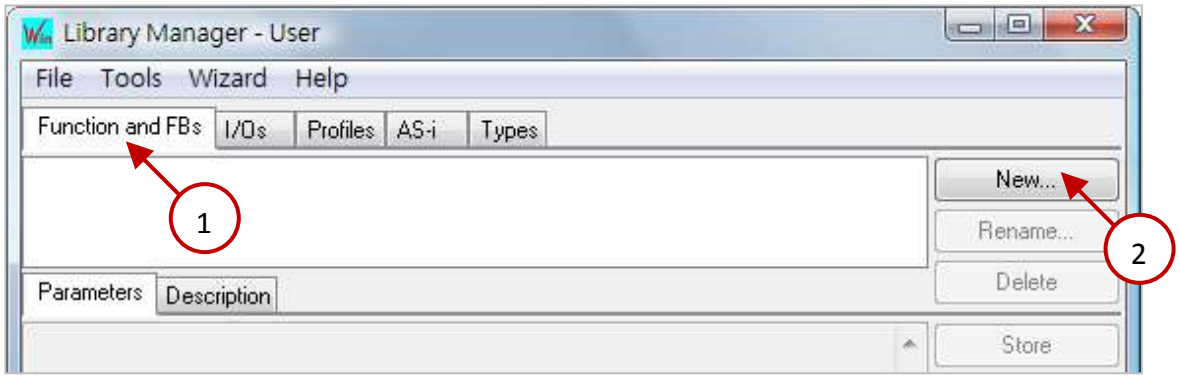
To begin, follow these steps:

1. Click All Programs → Win-GRAF → Libraries → OEM from the Start menu as the figure below.
   Note: If the Win-GRAF Workbench is opened, the user cannot add or edit Win-GRAF Library.

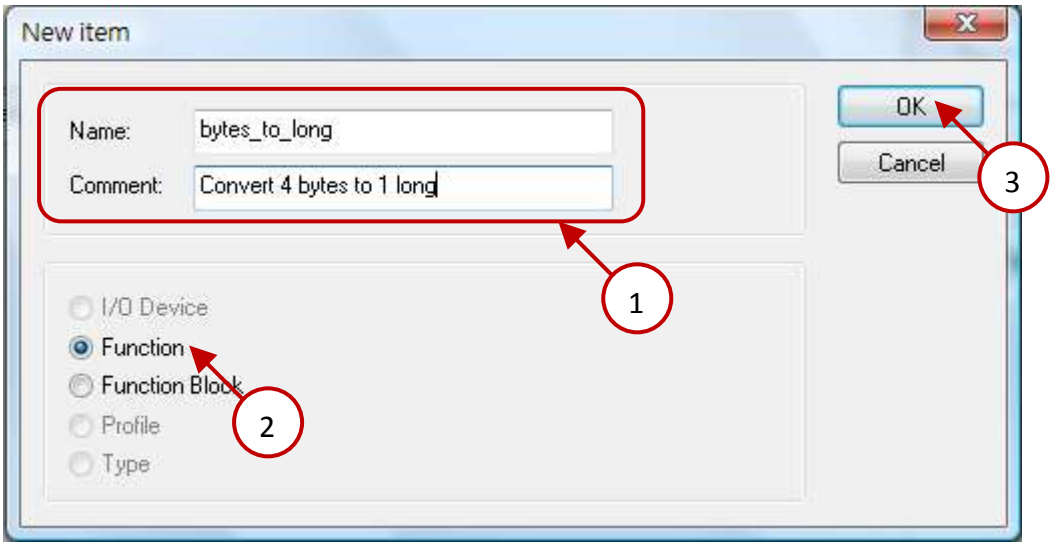
2. Create a library group named "User" in order to easy maintenance and management.



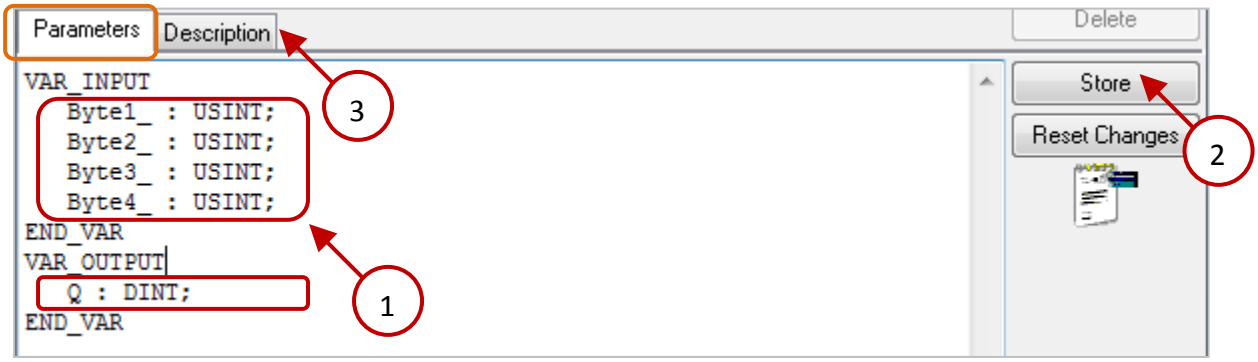
3. Select the "Function and FBs" tab and click the “New” button.



4. Enter the name and comment, and select the type as a Function and then click the “OK” button.

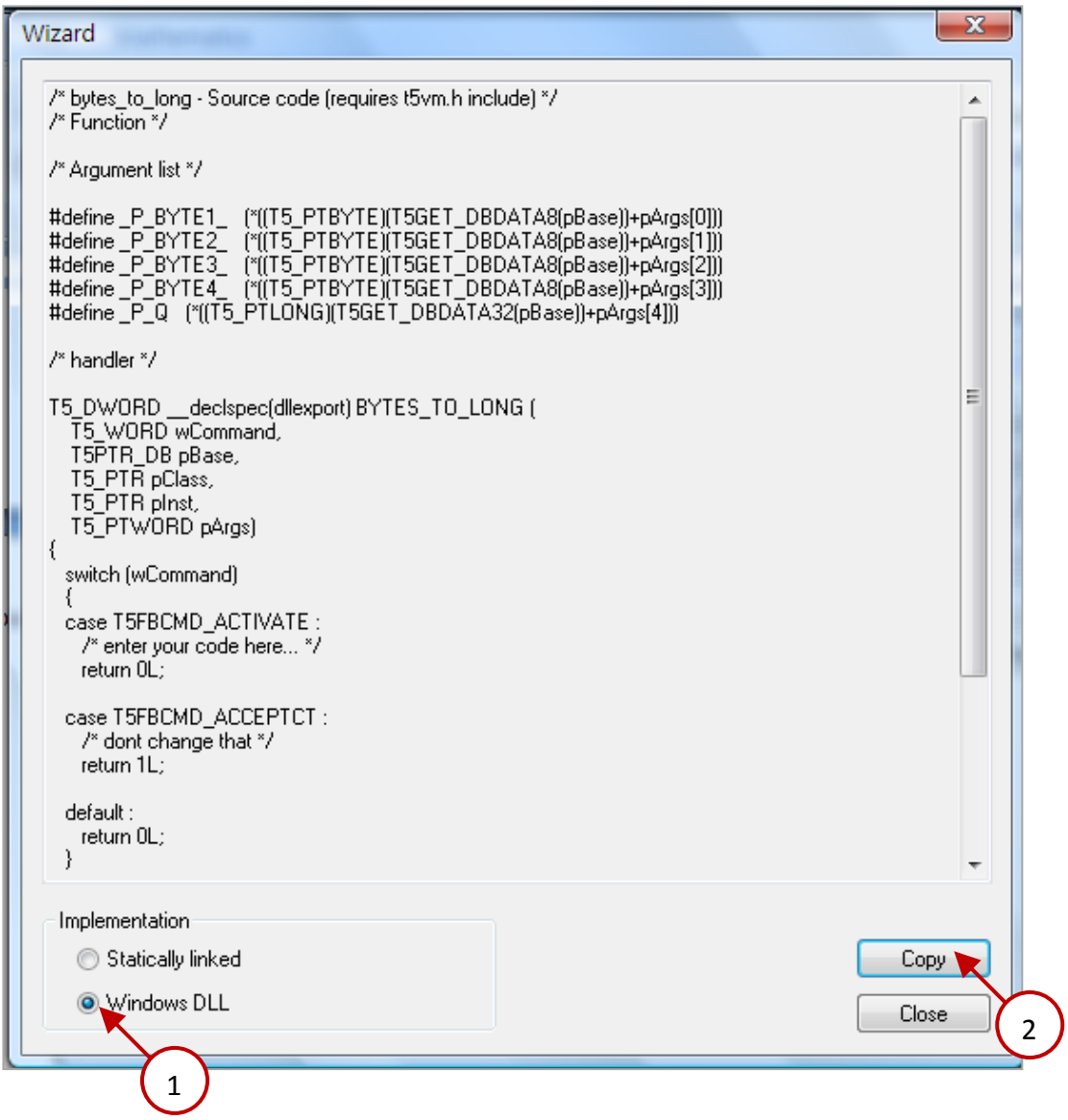


5. Declare the prototype of this Function.
   The user can declare parameters like this way - “variable name : data type;”, and refer Appendix A for the data type. The content between the "VAR_INPUT" and "END_VAR" are passed-in parameters; the content between the "VAR_OUTPUT " and "END_VAR" is returned parameter. After completing it, click “Store” to save. Then, click “Description” tab to add the technical notes for this Function and then click “Store” to save.

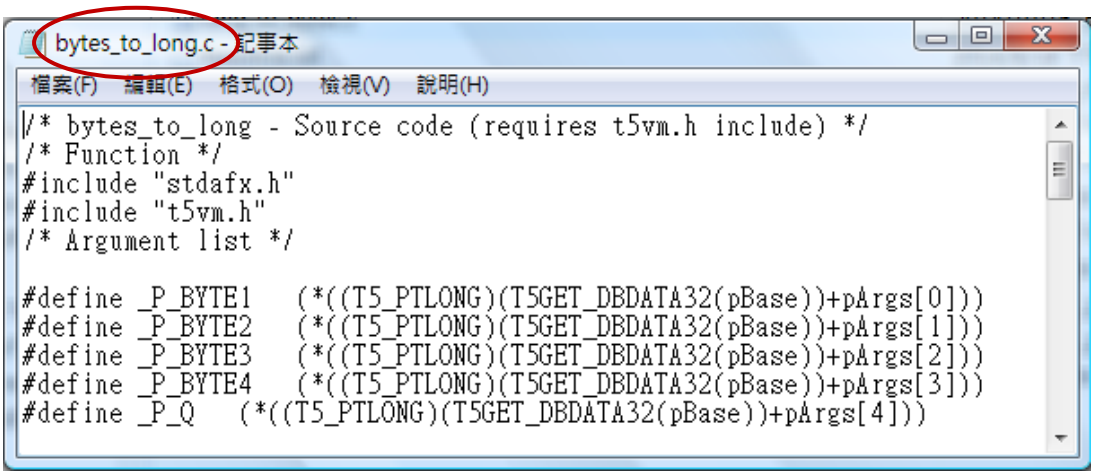


6. Click “Wizard” → “Function/Block Source Code” to generate the source code of this Function.

7. Select “Windows DLL” and click “Copy” button to copy the source code.



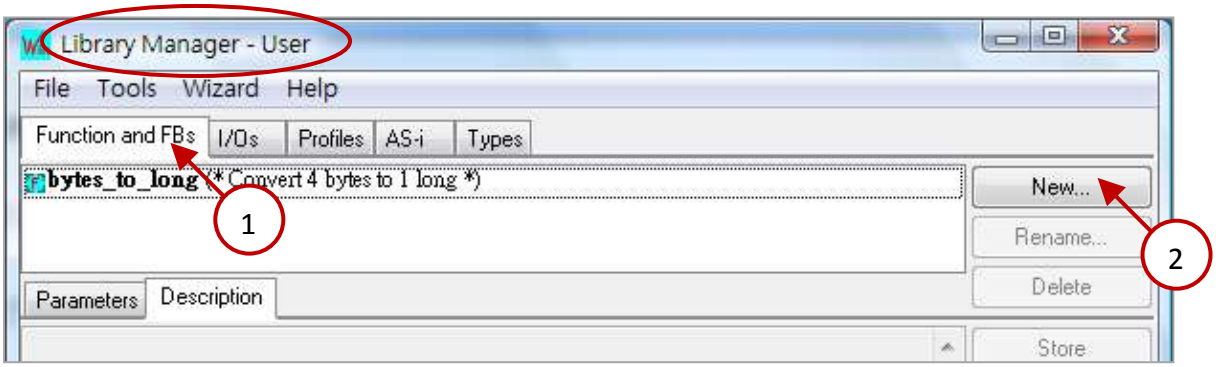
8. Paste the source code into the text editor (e.g., Notepad) and save it as “bytes_to_long.c”.



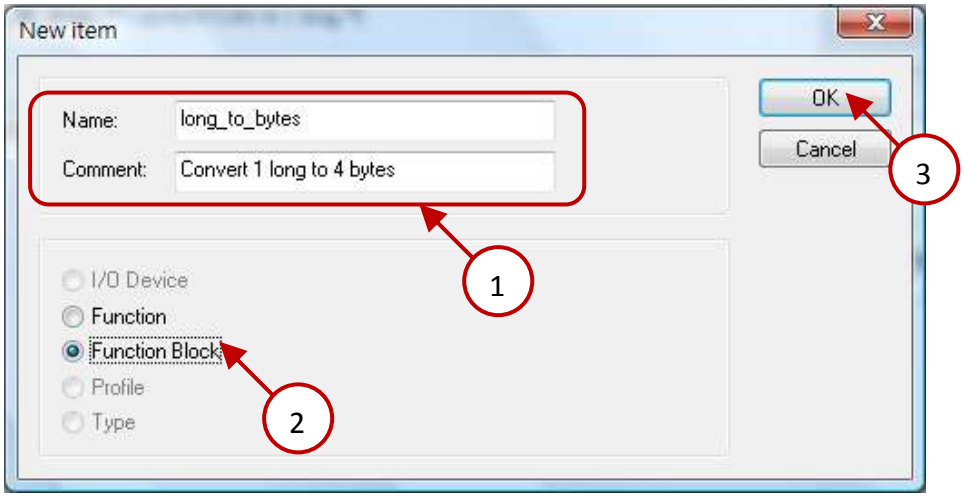
18.3.2 Define Function Block Lib

This section uses the similar way as previously described to define the Win-GRAF library of the Function Block (e.g., "long_to_bytes") and then create a file of source code (e.g., "long_to_bytes.c").

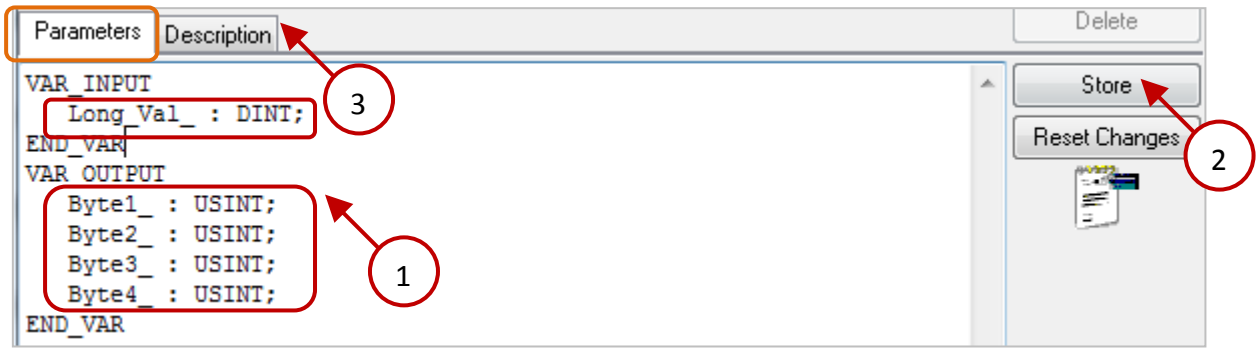
1. Refer the Previous Section to open "Library Manager" (All programs → Win-GRAF → **Libraries → OEM**) and open the library group - "User", and then click "New" button to add a Function Block.



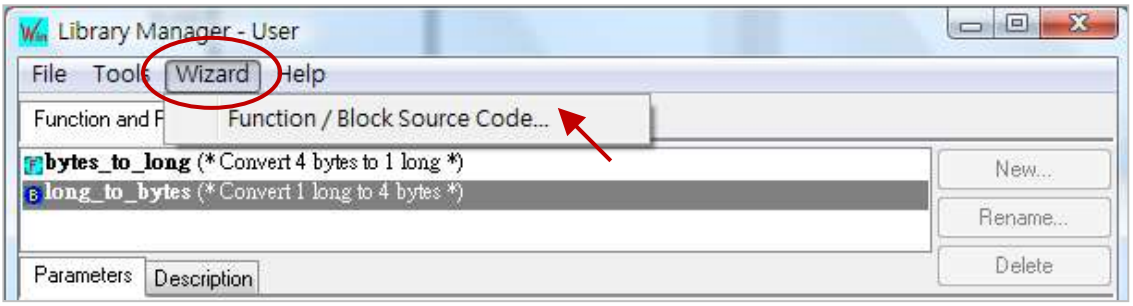
2. Enter the name and comment, and select the "Function Block" type and then click the "OK" button.



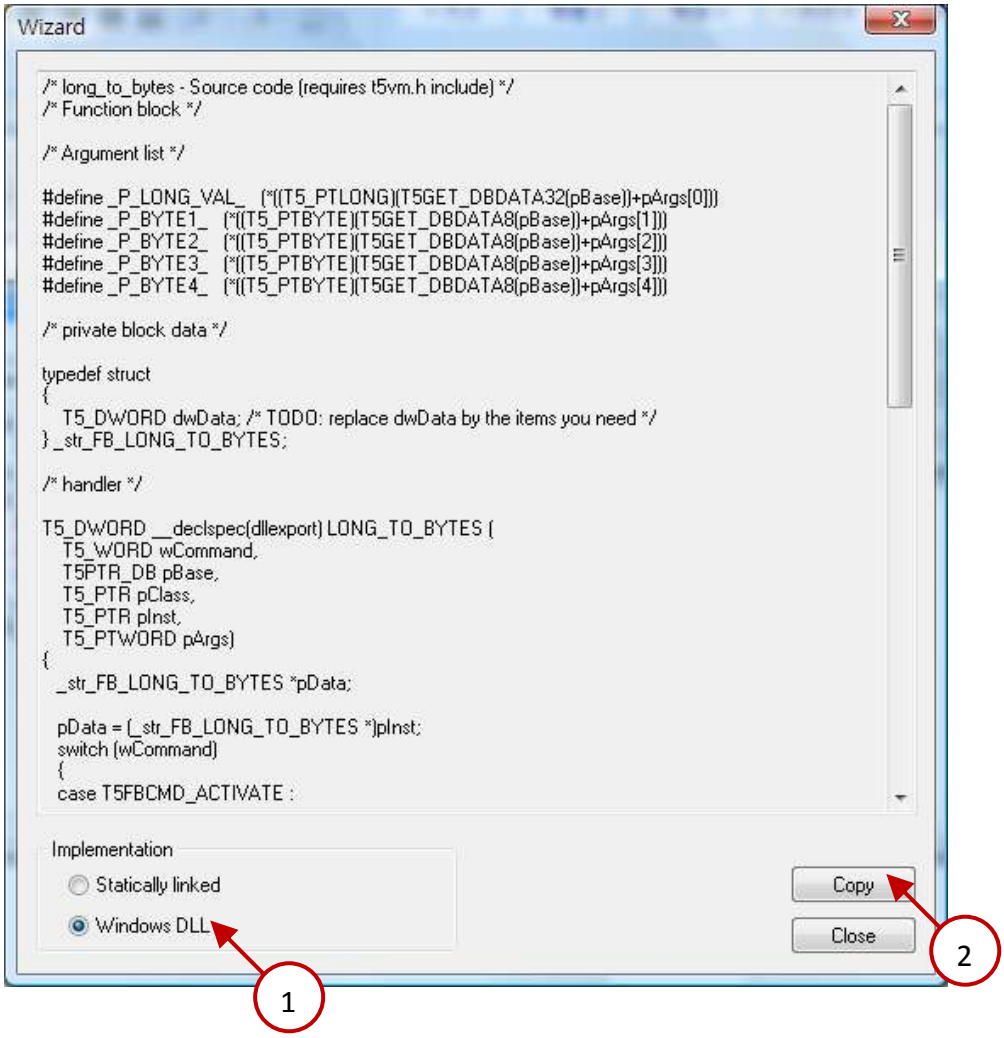
3. Declare the prototype of this Function Block.
   The user can declare parameters like this way - "variable name : data type;", and refer Appendix A for the data type. The content between the "VAR_INPUT" and "END_VAR" is passed-in parameter; the content between the "VAR_OUTPUT " and "END_VAR" are returned parameters.
   After completing it, click "Store" to save. Then, click "Description" tab to add the technical notes for this Function Block and then click "Store" to save.



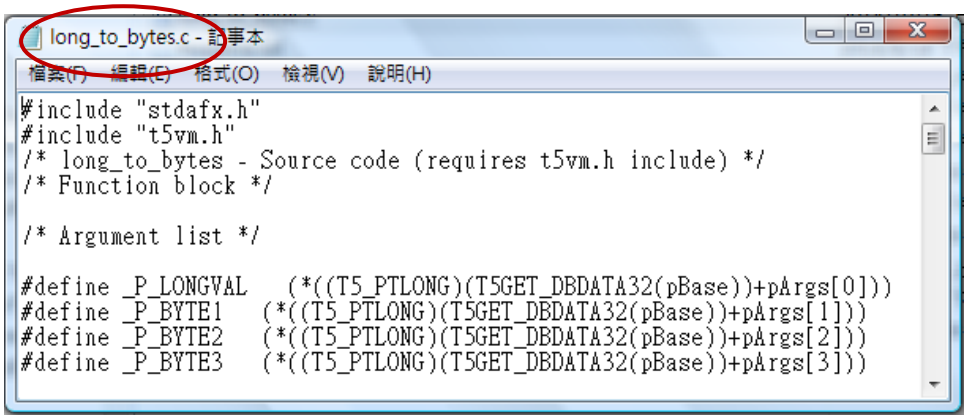
4. Click “Wizard” → “Function/Block Source Code” to generate the source code of this Function Block.



5. Select “Windows DLL” and click “Copy” button to copy the source code.



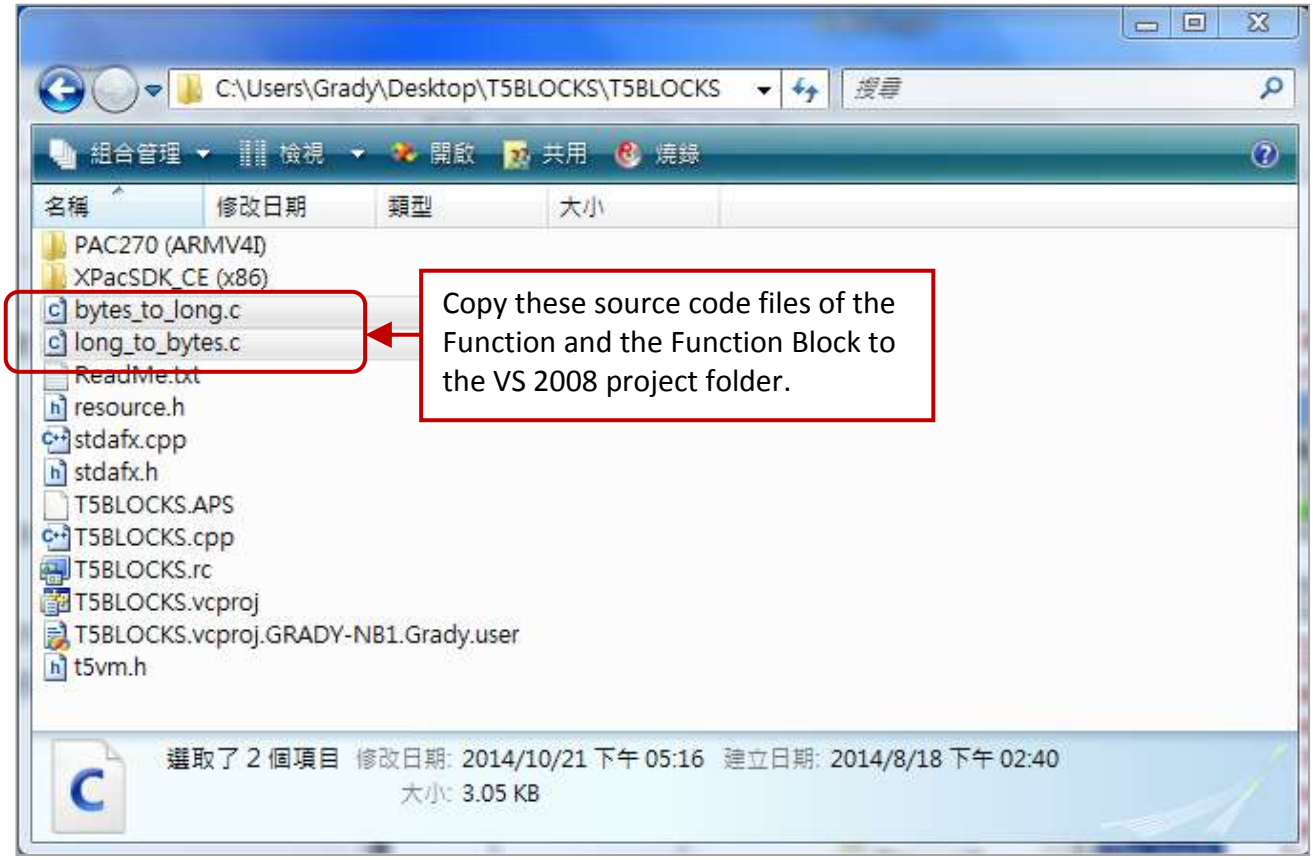
6. Paste the source code into the text editor (e.g., Notepad) and save it as “bytes_to_long.c”.



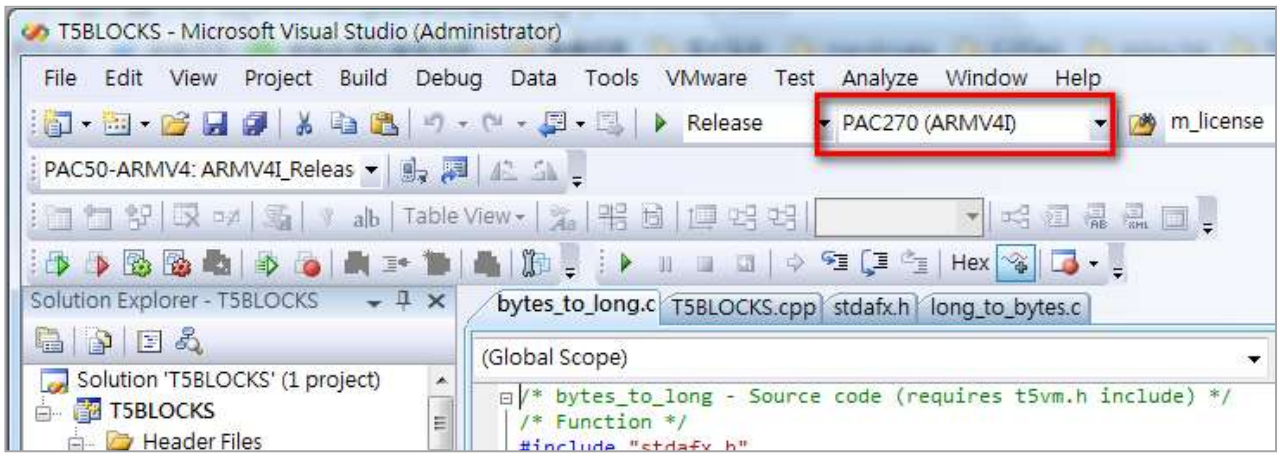
18.4 Edit the Logic of the Function and Function Block

Note: Please make sure your PC has installed the Visual Studio 2008 and WinPAC/XPAC SDK before operating the following steps.

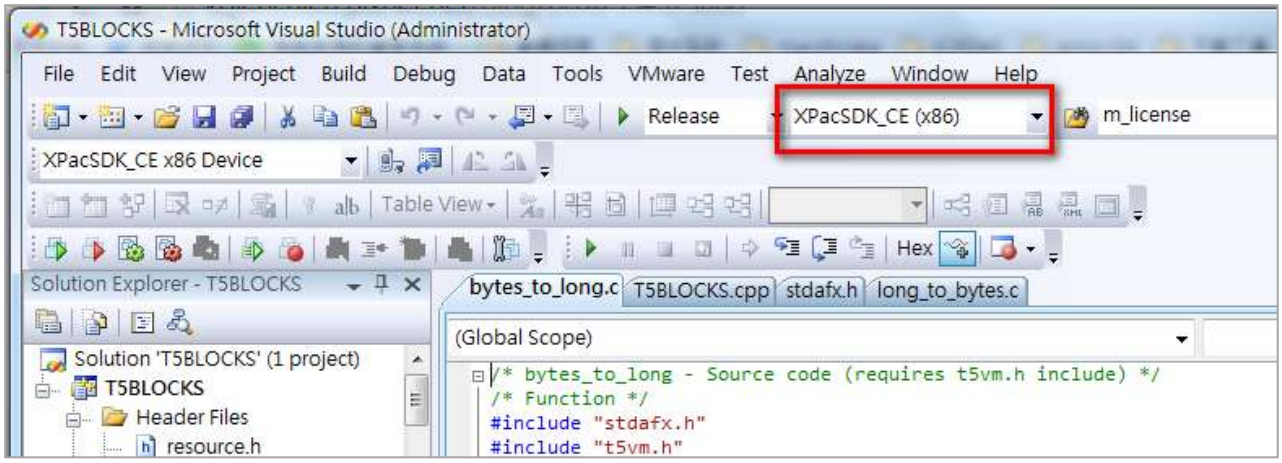
1. As the figure below, copy these source code files of the Function and the Function Block (described in Section 18.3) to the VS 2008 project folder on your PC. (Or, get the VS 2008 sample project folder from the CD-ROM: \napdos\Win-GRAF\demo-project\user_c_lib\demo_user_c)



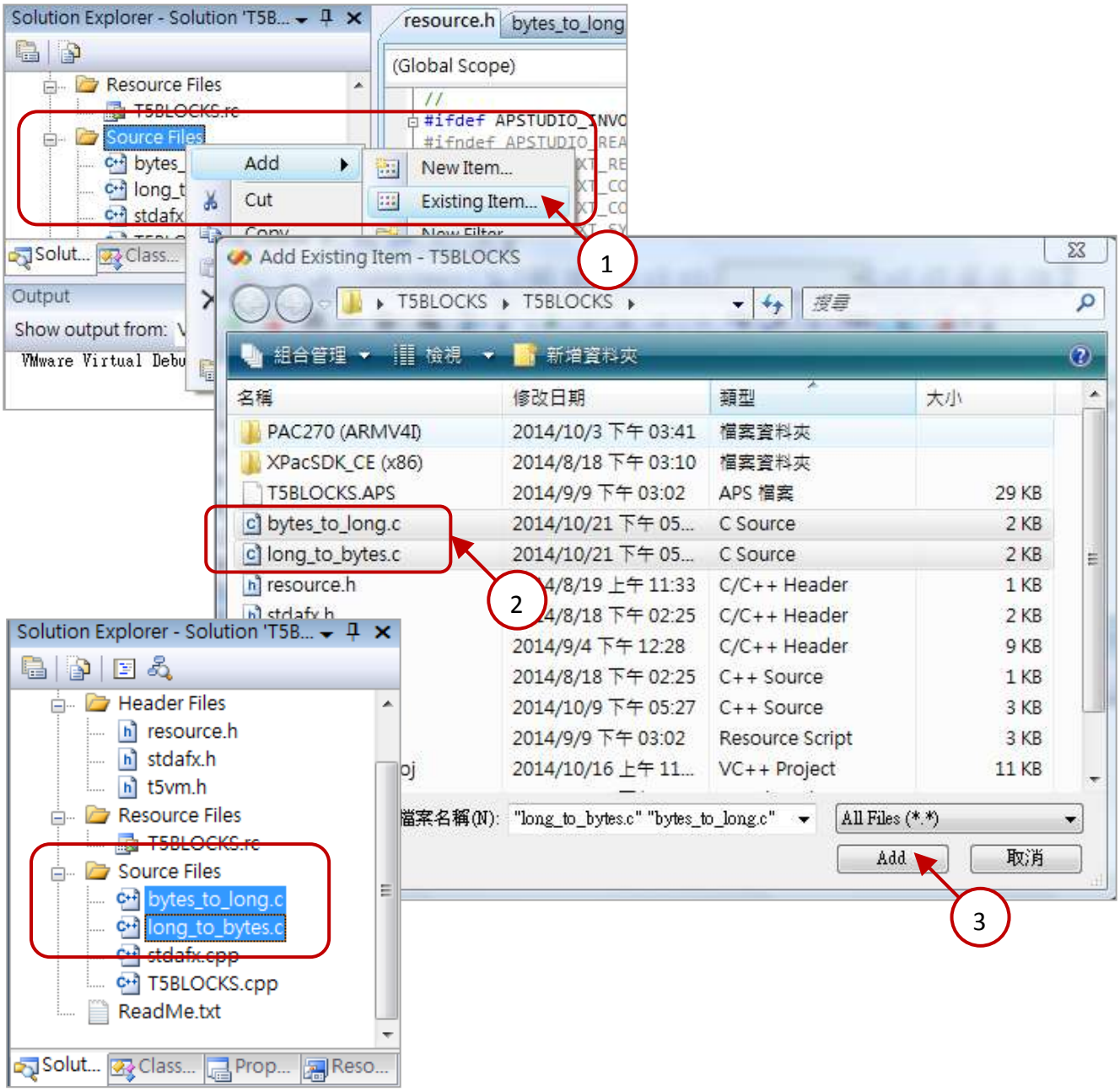
2. Then, make sure your VS 2008 project settings are correct for your PAC.
   (**Note:** the settings are different between different controllers.)
   For WP-8xx8, WP-8xx8-CE7, VP-x2x8-CE7, and WP-5xx8-CE7, it must set to "PXA270 (ARMV4I)".



For XP-8xx8-CE6, it must set to "XPacSDK (x86)".

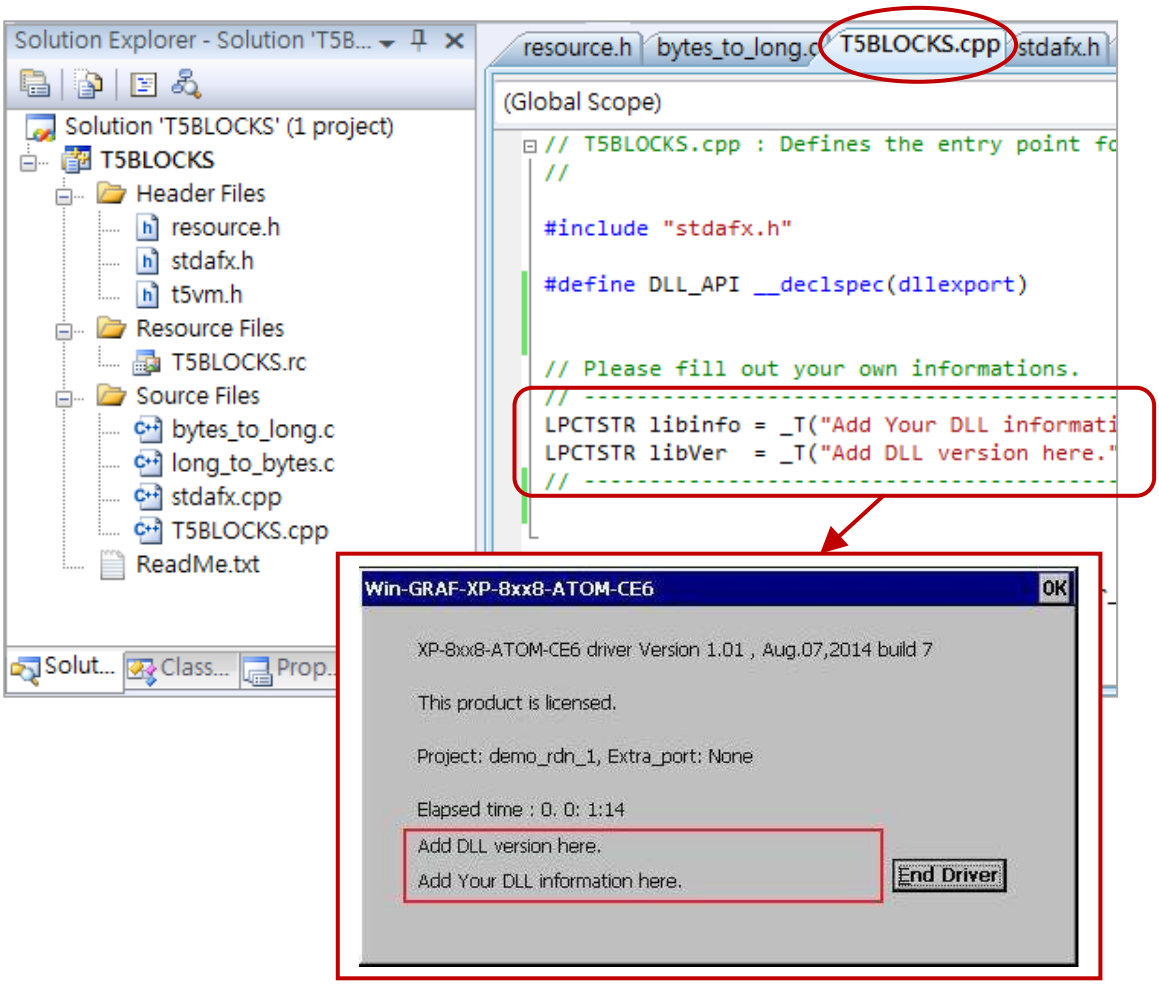


3. Here, we will add C source files of the Function and Function Block to the VS 2008 project. Mouse right-click on the "Source Files" folder and click “Add” → “Existing Item…”, select the previously copied C source files and click the “Add” button.

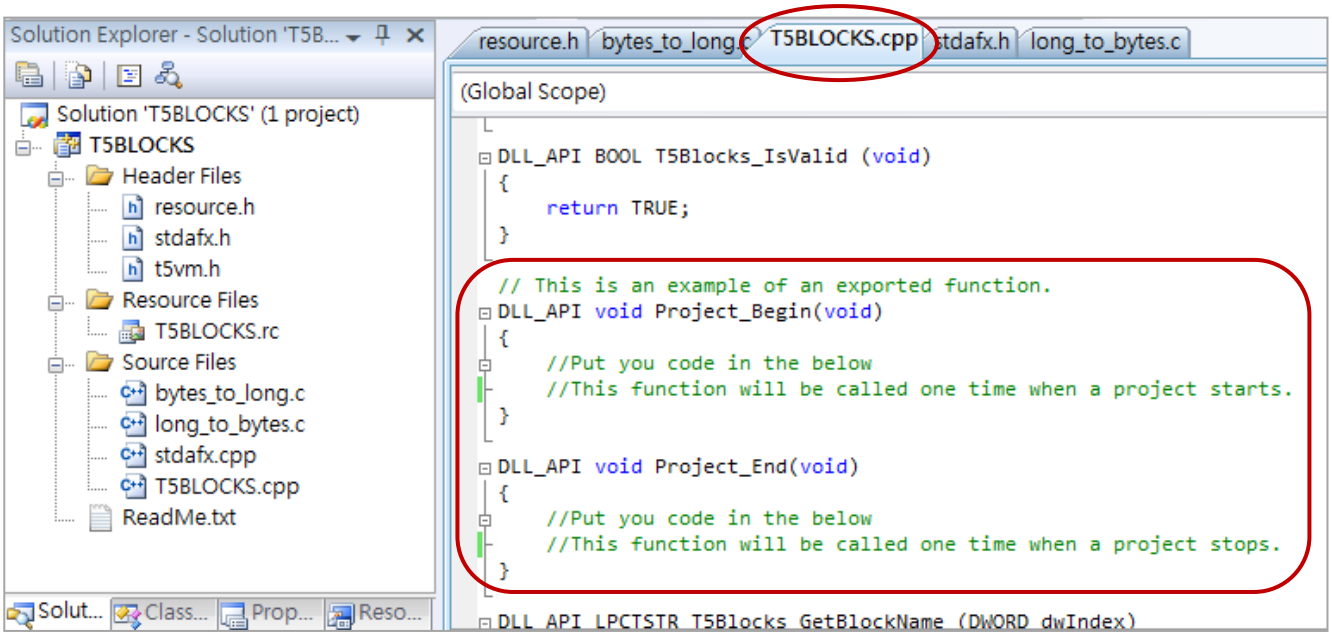


18.4.1 Edit the "T5BLOCKS.cpp"

Fill out your "libinfo" and "libVer" information in the "T5BLOCKS.cpp" file. The information will show on the Win-GRAF driver dialog on the PAC (as the figure below).

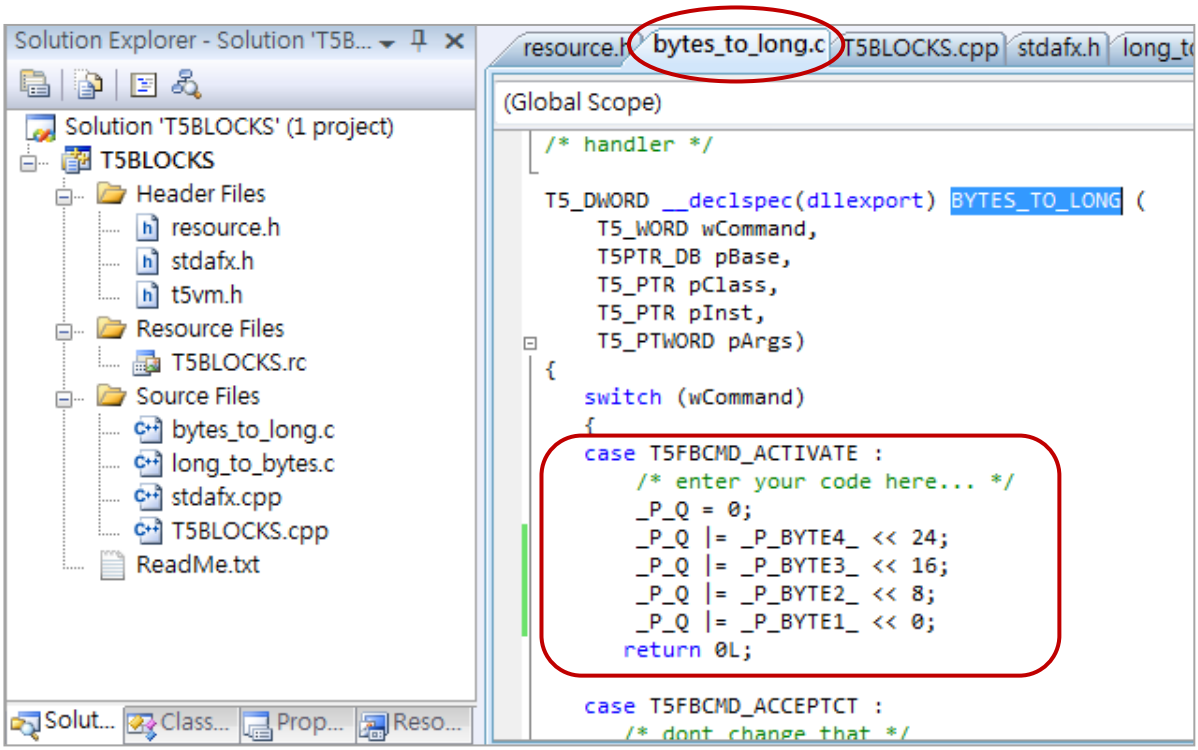


If you have some operations to process when a new Win-GRAF project starts or stops. Please edit the "Project_Begin" and "Project_End" functions in the "T5BLOCKS.cpp" file.



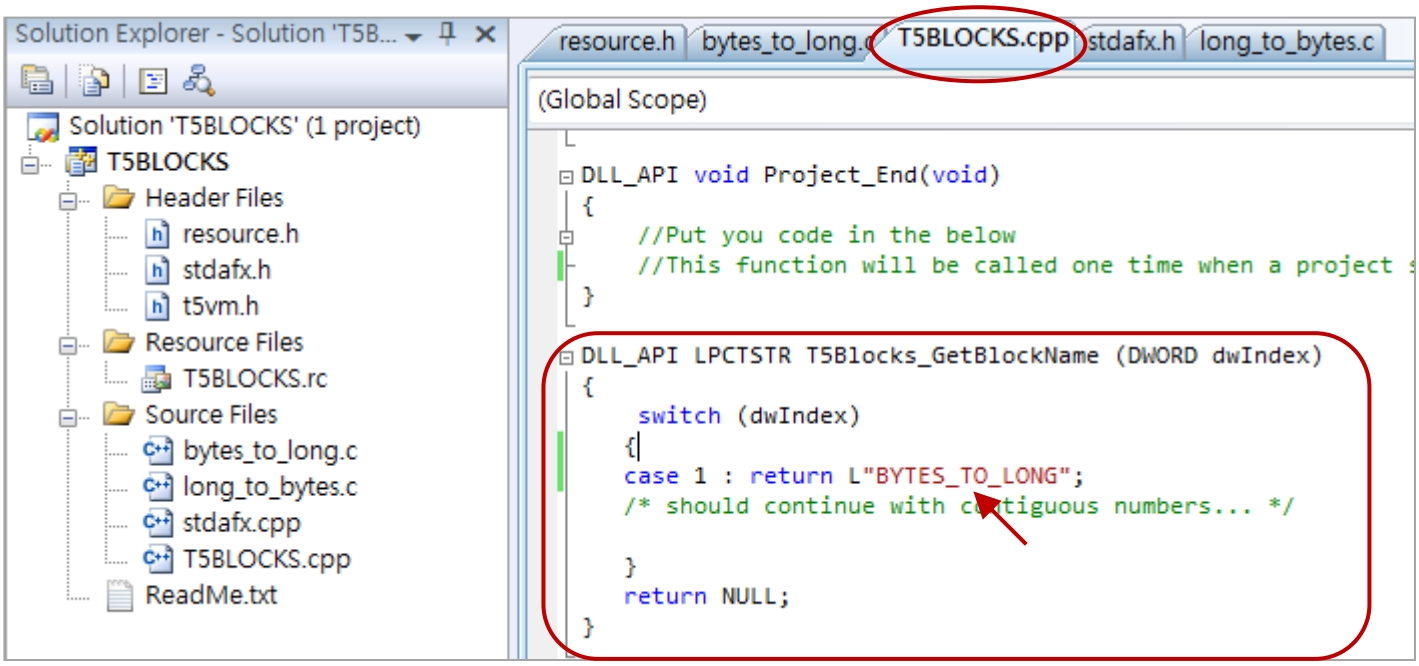
18.4.2 Edit the Logic of the Function (In this example is “bytes_to_long. c”)

First, add your logical expression for this Function in the switch case statement –"T5FBCMD_**ACTIVE**".



Then, add this Function name (e.g., “BYTES_TO_LONG”) into the switch case statement of the "T5Blocks_GetBlockName" functions in the “T5BLOCKS.cpp” file.

Note: The number of the case label (e.g., case 1) must start from "1" and continue with contiguous numbers.



18.4.3 Edit the Logic of the Function Block (In this example is “long_to_bytes.c”)

```
typedef struct
{
    T5_DWORD dwData; /* TODO: replace dwData by the items you need */
} _str_FB_LONG_TO_BYTES;

T5_DWORD __declspec(dllexport) LONG_TO_BYTES (
    T5_WORD wCommand,
    T5PTR_DB pBase,
    T5_PTR pClass,
    T5_PTR pInst,
    T5_PTWORD pArgs)
{
   _str_FB_LONG_TO_BYTES *pData;

   pData = (_str_FB_LONG_TO_BYTES *)pInst;
   switch (wCommand)
   {
   case T5FBCMD_ACTIVATE :
      /* activates the function block */
      /* enter your code here... */
      _P_BYTE1_ = _P_LONG_VAL_ & 0xFF;
      _P_BYTE2_ = (_P_LONG_VAL_ >> 8) & 0xFF;
      _P_BYTE3_ = (_P_LONG_VAL_ >> 16) & 0xFF;
      _P_BYTE4_ = (_P_LONG_VAL_ >> 24) & 0xFF;
      return 0L;

   case T5FBCMD_INITINSTANCE :
      /* initialize private data */
      /* enter your code here... */
      return 0L;

   case T5FBCMD_EXITINSTANCE :
      /* release private data */
      /* enter your code here... */
      return 0L;

   case T5FBCMD_HOTRESTART :
      /* actuate private data for hot restart */
      /* enter your code here... */
      return 0L;

   case T5FBCMD_SIZEOFINSTANCE :
      /* dont change that */
      return (T5_DWORD)sizeof(_str_FB_LONG_TO_BYTES);

   case T5FBCMD_ACCEPTCT :
      /* dont change that */
      return 1L;

   default :
      return 0L;
   }
}
```

Each used Function Block in the Win-GRAF project has distributed a “private structure”memory for users to use.

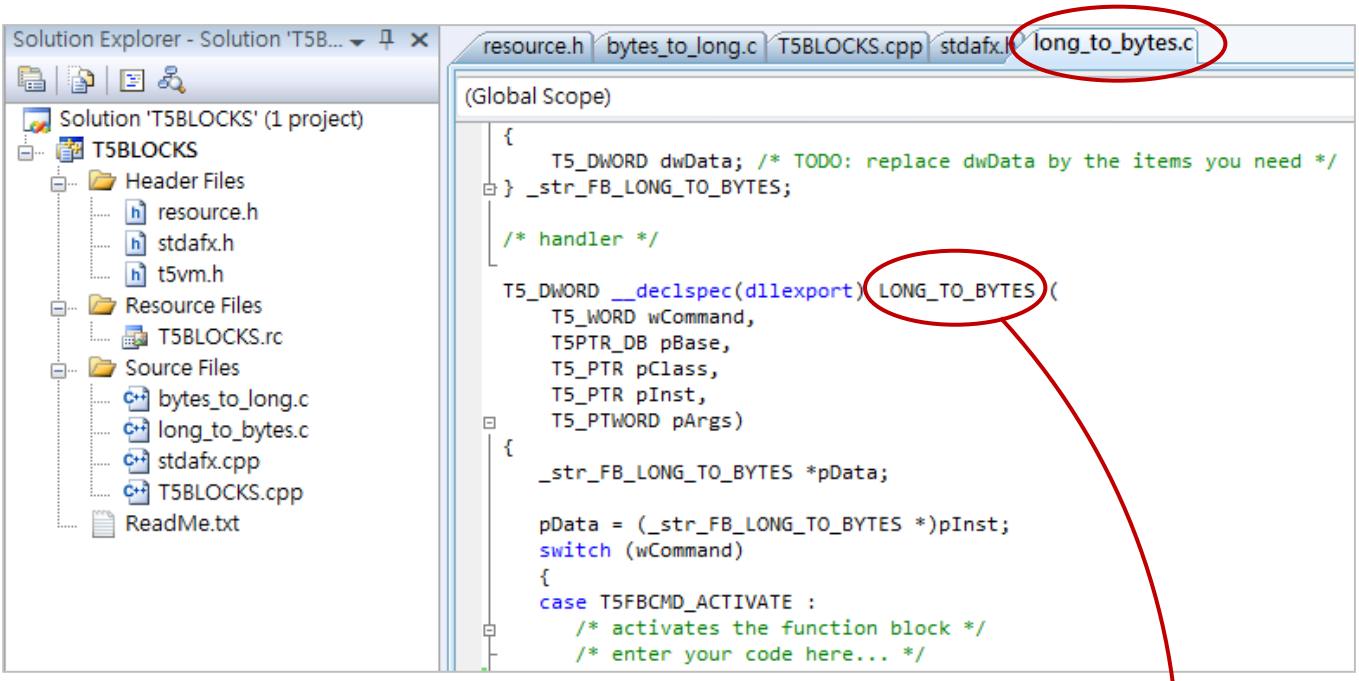
Doing expressions in each Cycle.

It used to initialize the “private structure” when the project starts.

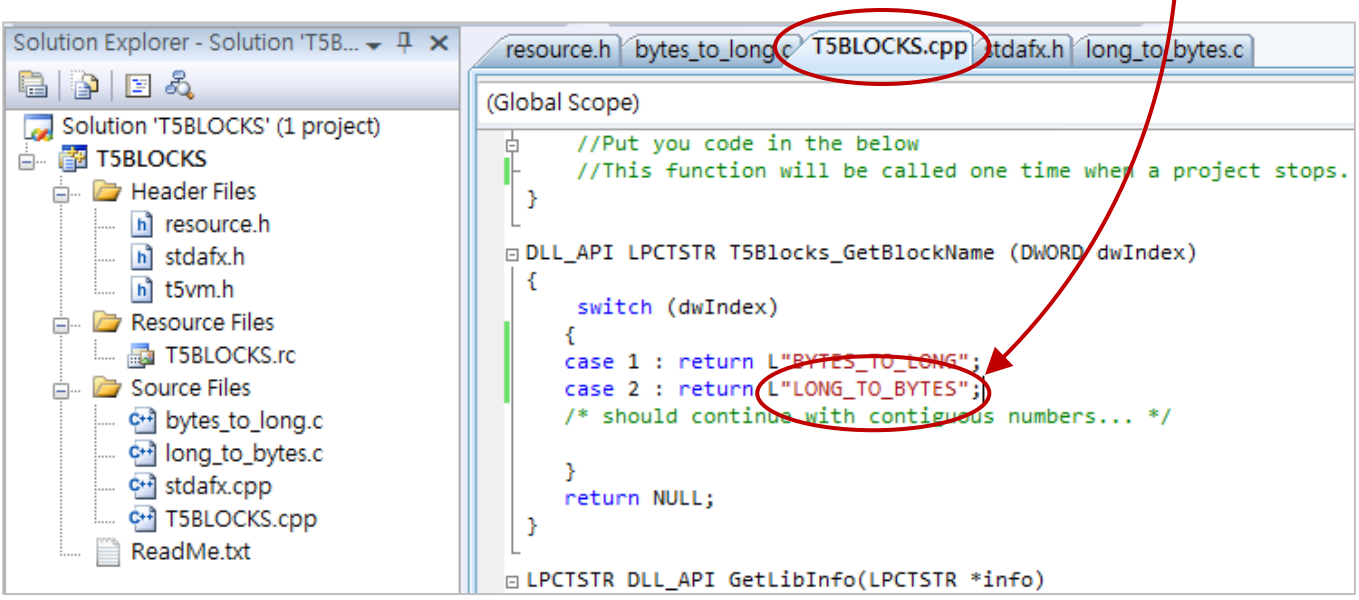
It used to close the “private structure” when the project stops.

Update data in the “private structure” memory when the project hot re-tarts or on-line change.

Finally, remember to add the Function Block name (e.g., “LONG_TO_ BYTES”) into the switch case statement of the "T5Blocks_GetBlockName" functions in the “T5BLOCKS.cpp” file.

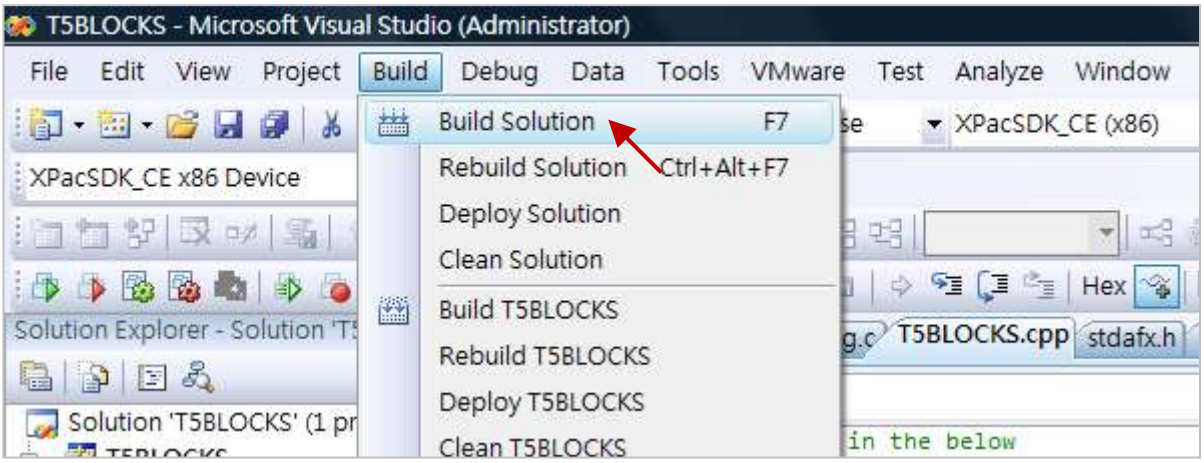


Note: The number of the case label (e.g., case 1) must start from "1" and continue with contiguous numbers.

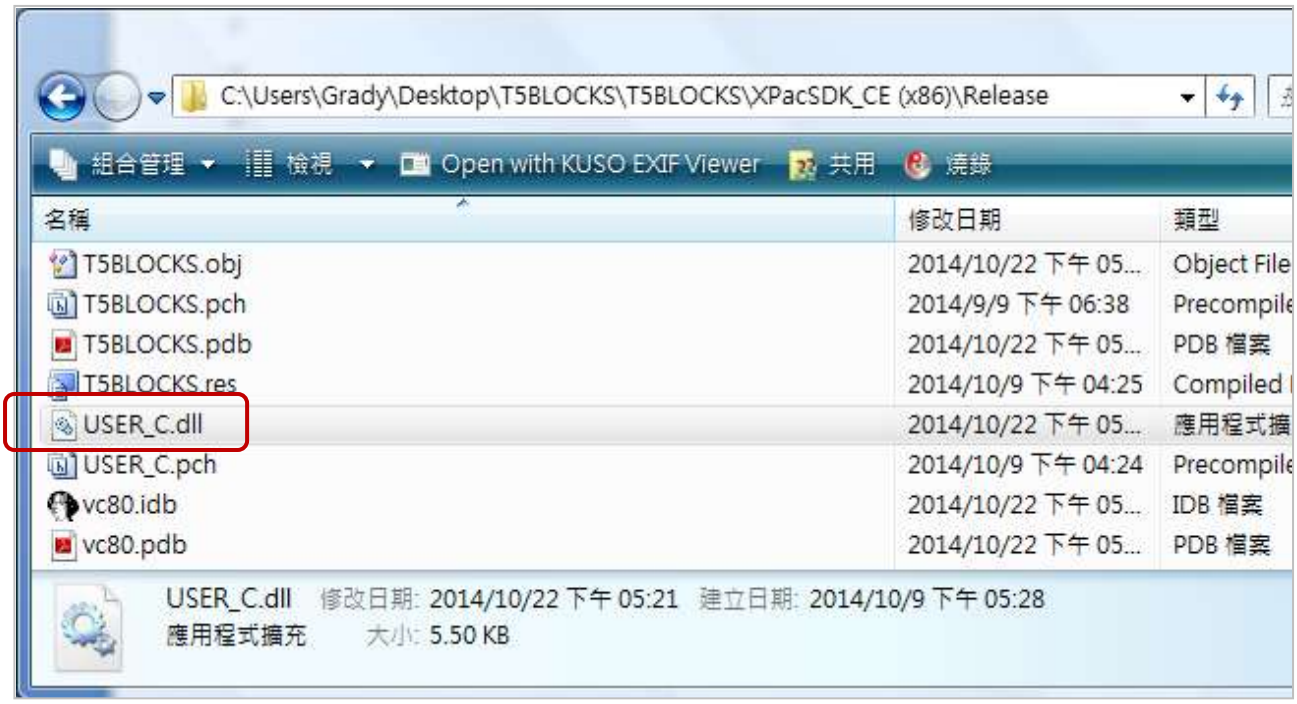


18.4.4 Trying to Compile the Project

1. Click "Build" > "Build Solution" to generate a DLL file.



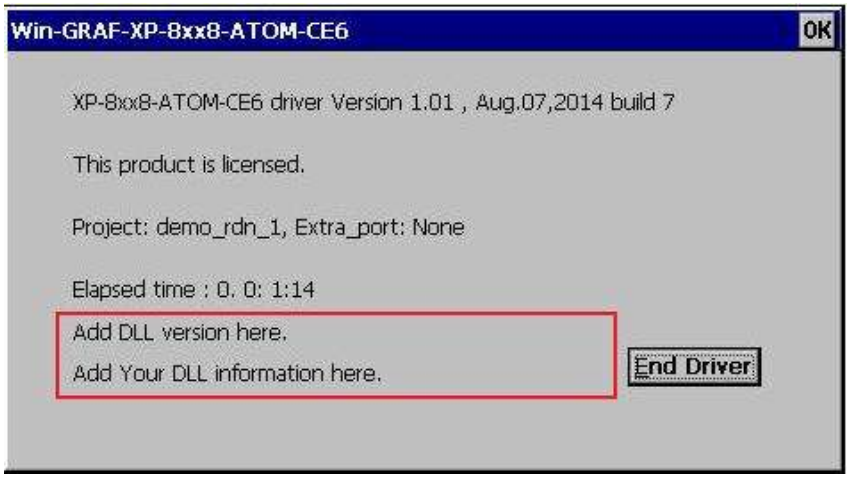
2. After a successful compilation, copy the “USER_C.dll” file to the path "\System_disk\Win-GRAF\" in your PAC, and then reboot it.



18.5 Test your own Function and Function Block

1. Copy the "user_c.dll" file to the same folder as the Win-GRAF driver (i.e., \System_disk\Win-GRAF\) on the PAC by using FTP. And, reboot the PAC.

   If a proper DLL file is detected by the Win-GRAF Driver, its dialog will show as below.



2. Open the Win-GRAF project that includes your own Function and Function Block, and then compiles and download this project to the Win-GRAF PAC.

In addition, there are some available files in the Win-GRAF PAC's CD-ROM for users to test.

(1) WP-8xx8, WP-8xx8-CE7, VP-x2x8-CE7 and WP-5xx8-CE7:
\napdos\Win-GRAF\demo-project\user_c_lib\wp_vp\user_c.dll

(2) XP-8xx8-CE6:
\napdos\Win-GRAF\demo-project\user_c_lib\xpc\user_c.dll

(3) Copy the Win-GRAF Library folder -"User" to the following path on your PC.
C:\Win-GRAF\DATA\HWDEF\

(4) Open the Win-GRAF sample project – "demo_user_c.zip", and then compiles and download this project to the Win-GRAF PAC. (Refer the Section 13.1 to open the Win-GRAF project from a zip).

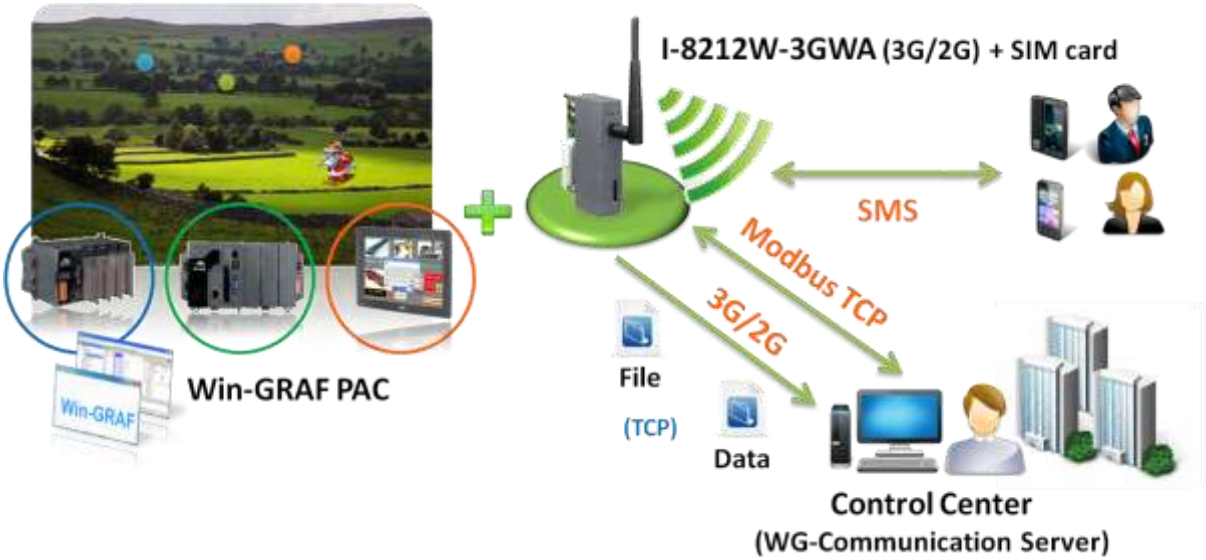
Chapter 19 Using 3G Modules - I-8212W-3GWA

Note:

1. Due to the product certification issue, the I-8212W-3GWA module can sales in certain areas. Please contact our agents for more information.
2. There is one another 3G Solution, that is, the user can buy a 3G Router come with a SIM card. By this way, the Win-GRAF PAC can also connect to the Internet via the 3G network.

Sending back the collected data to the control center is necessary in some application. However, there may be no cable can reach the field or the cost of the network wiring is too expensive. ICP DAS released the "Win-GRAF PAC + I-8212W-3GWA" solution for such applications. Designers can collect I/O data or other application data by program a PLC application (Ladder, ST, Function block, etc.) with Win-GRAF software. Using the device – "I-8212W-3GWA" (insert the SIM card inside that has registered the 3G/2G service from the Telecom Company) to connect Internet by dial-up 3G/2G, then the PLC can send TCP data to the center.

3G/2G Wireless Application



The following Win-GRAF driver version supports the dial-up 3G/2G access with the I-8212W-3GWA.

XP-8xx8-CE6 : 1.03 or later WP-8xx8 : 1.05 or later VP-x2x8-CE7 : 1.01 or later

If the Win-GRAF driver version of your PAC is older than the above listed version, please visit the http://www.icpdas.com/root/product/solutions/softplc_based_on_pac/win-graf/download/win-graf-driver.html to download the newer driver.

I-8212W-3GWA: http://m2m.icpdas.com/i-8212w-3GWA.html

19.1 Hardware Installation

The I-8212W-3GWA supports 3G/2G wireless communication. Insert the 3G SIM card (that registered the 3G/2G function from the Telecom Company) into the "SIM card" socket of this 3G/2G module and make sure the antenna has installed well.

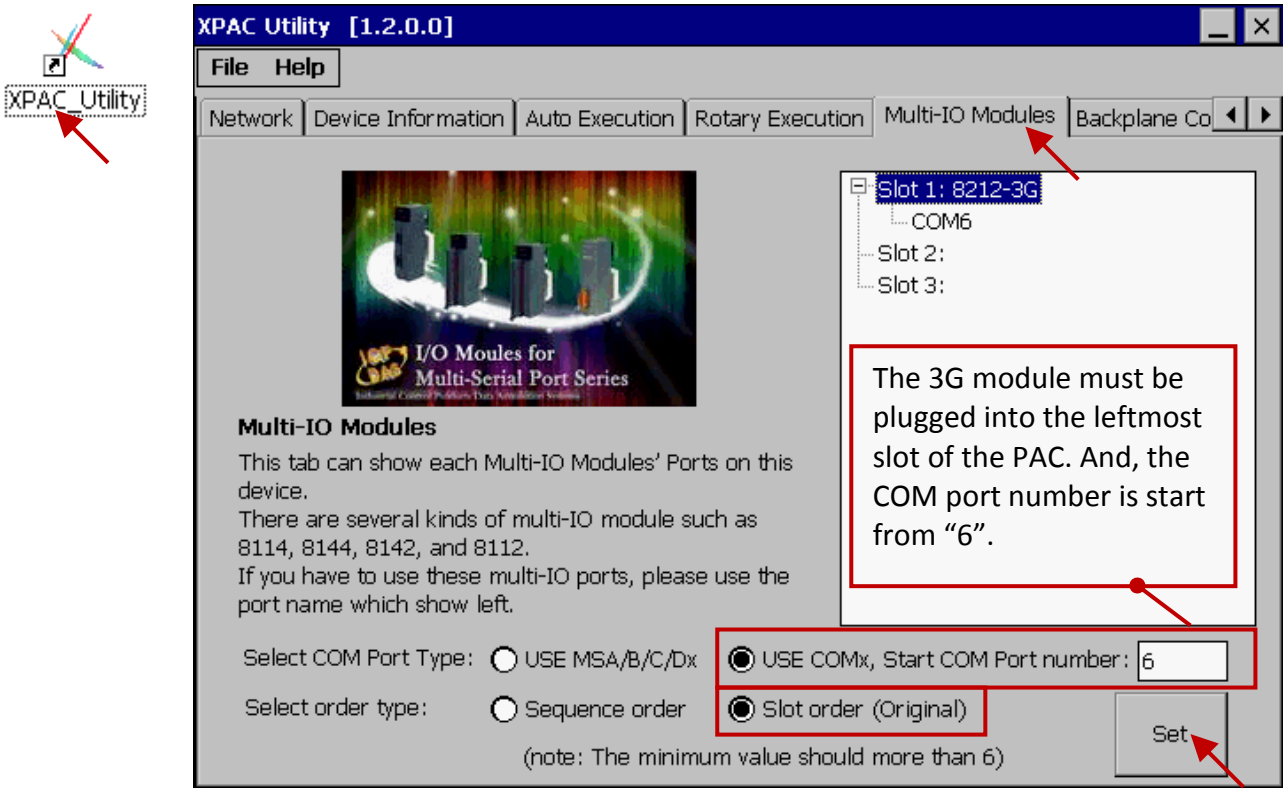
If your PAC is **XP-8xx8-CE6**, plug the I-8212W-3GWA in its slot **1**.
(I.e., the leftmost I/O slot of the XPAC).
If your PAC is **WP-8xx8 or VP-x238-CE7**, please plug the I-8212W-3GWA in its slot **0**.
(I.e., the leftmost I/O slot of the WinPAC. The slot number of ViewPAC is shown on the back).

Then power on the PAC and run PAC Utility (for example, run "XPAC_Utility" for the XPAC) to set up the "COM6" port of the I-8212W-3GWA. Remember to run "File > Save and Reboot" once to save the settings.

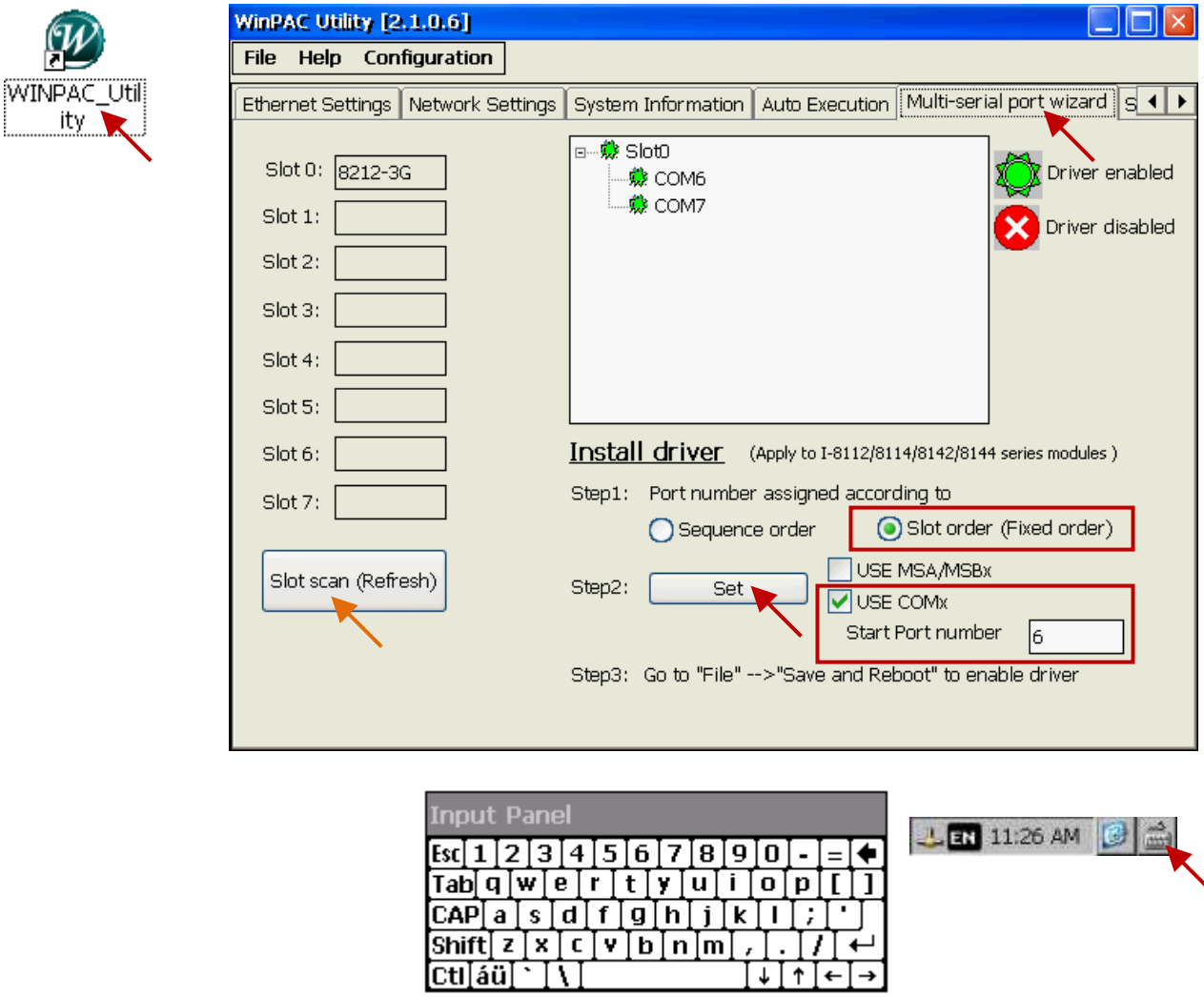
The user can find out the PAC Utility on the desktop or the PAC folder (e.g., \System_Disk\Tools\ XPAC_Utility), or download the Utility on FTP:

XPAC: ftp://ftp.icpdas.com/pub/cd/xp-8000-ce6/system_disk/tools/
WinPAC: ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/system_disk/tools/
ViewPAC: ftp://ftp.icpdas.com/pub/cd/winpac/napdos/vp-4000_ce50/system_disk/tools/

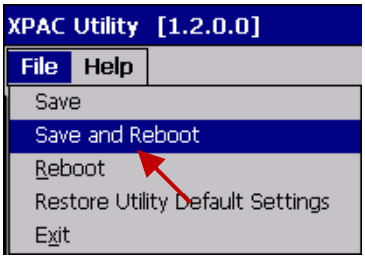
XPAC_Utility:



WinPAC_Utility:



Finally, run “File > Save and Reboot” in the PAC Utility.



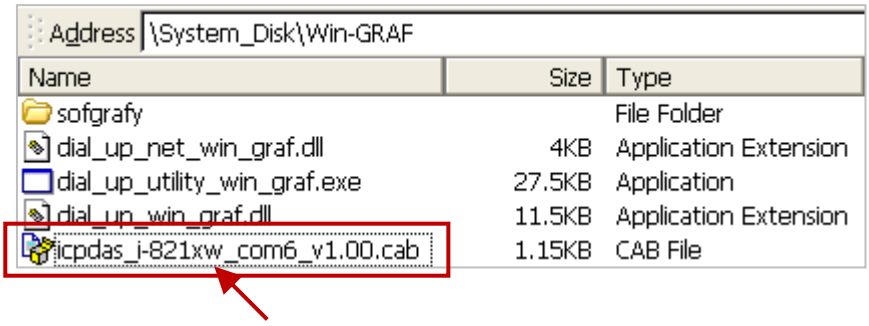
19.2 Software Installation

Please check the Win-GRAF driver version for your PAC is the correct version that listed in the Section 19 (P19-1). If not, update it.

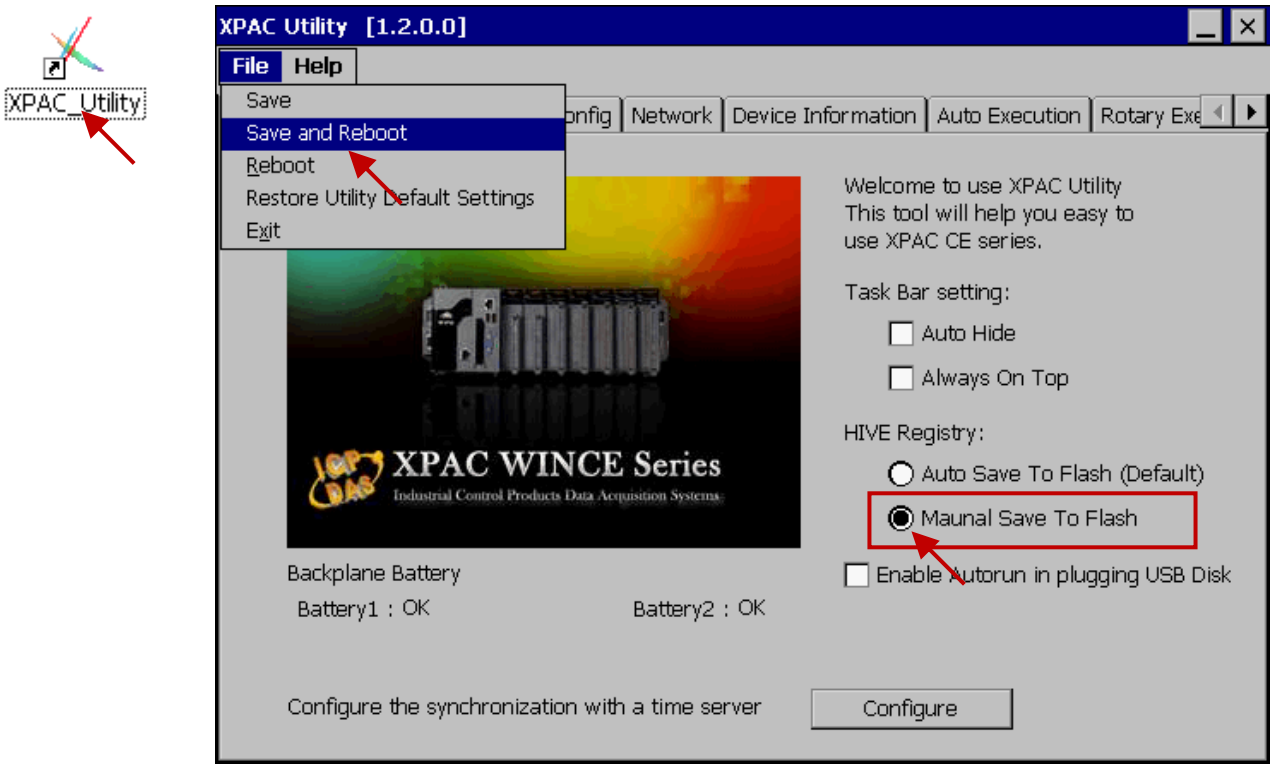
19.2.1 Install the I-8212W-3GWA (or I-8213-3GWA) Driver

Note: Plug the I-8212W-3GWA into the I/O slot**0** of the WinPAC or ViewPAC.
In addition, plug it into the I/O slot**1** of the XP-8xx8-CE6.

Double-click the "icpdas_i-821xw_com6_vx.xx.cab" file in the path of Win-GRAF PAC - **\System_Disk\Win-GRAF** to install the I-8212W-3GWA driver if the PAC is the XP-8xx8-CE6, WP-8xx8 or VP-x238-CE7.

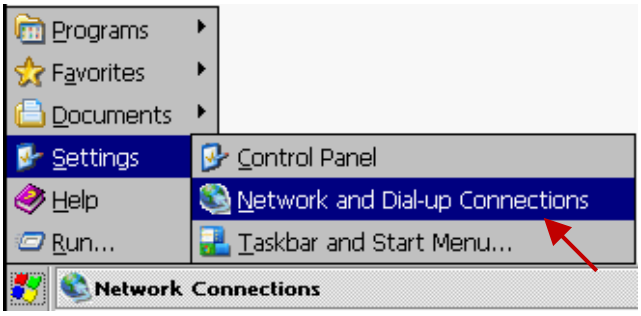


After completing the installation, remember to open the XPAC Utility (or WinPAC Utility, ViewPAC Utility) and run "File > Save and Reboot" to save the settings, then the PAC will restart automatically once.
(In the below figure, we use XP-8xx8-CE6 as a sample, select "Manual Save To Flash" and then run "File > Save and Reboot").

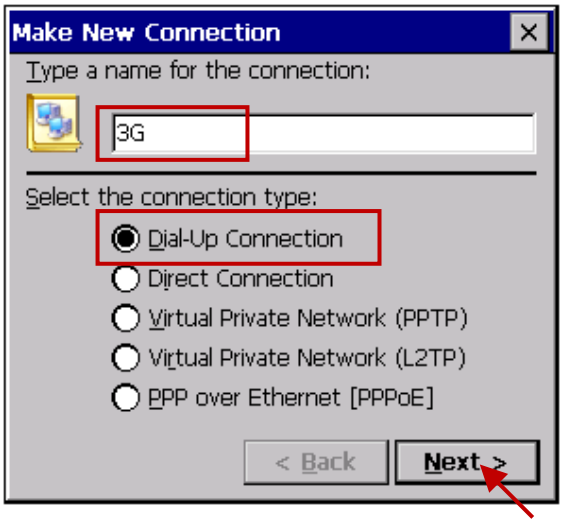


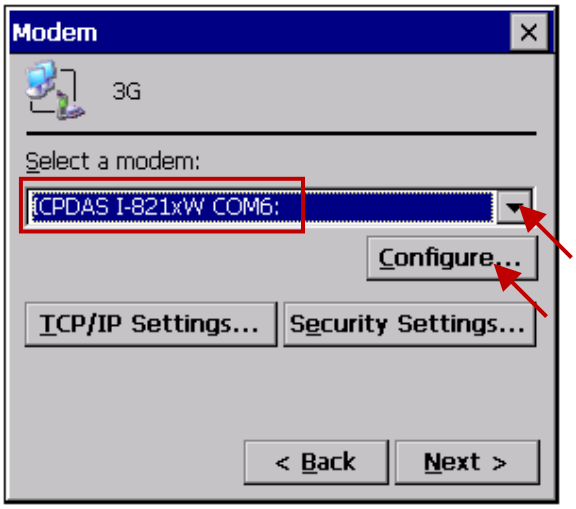
19.2.2 Configure the 3G/2G Dial-up Parameters

At first, get into the “Network and Dial-up Connections” and then run “Make New Connection” on the PAC.

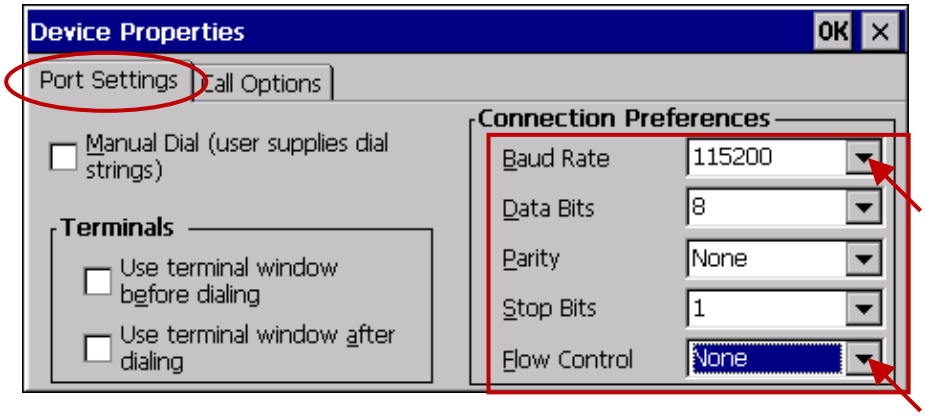


Select “Dial-Up Connection” and type an English name (e.g., “3G”, it allows to contain the numbers 0 to 9) and then click the “Next” button. Then, Select the modem - “ICPDAS I-821xW COM6:” and click the “Configure ...” button.





In the “Port Settings” tab, select “Baud Rate” as “115200”, “Data Bits” as “8”, “Parity” as “None”, “Stop Bits” as “1” and “Flow Control” as “None”.



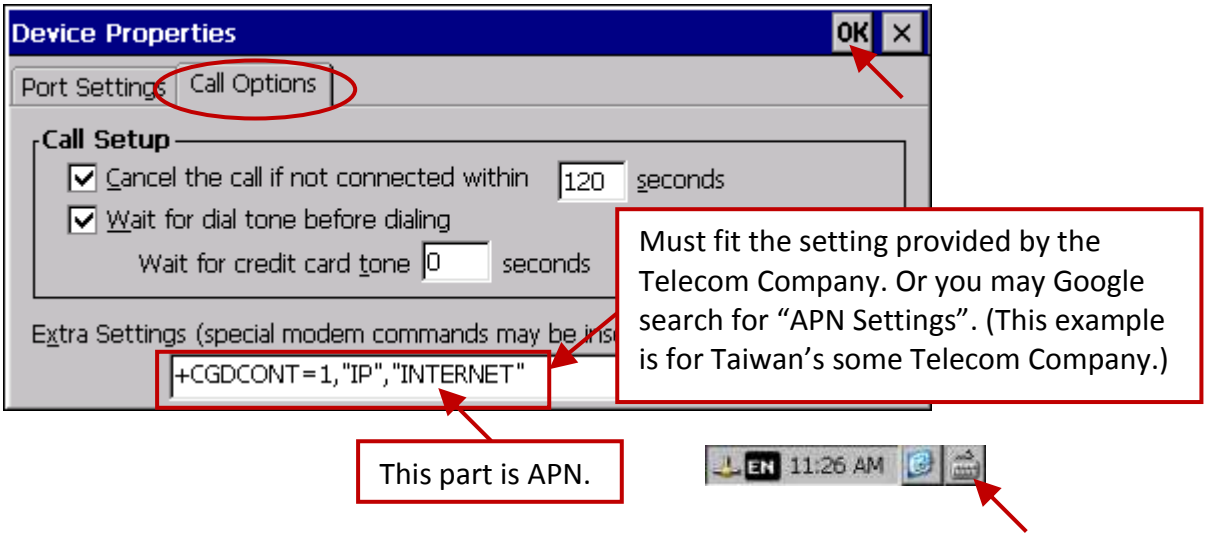
Then, click “Call Options” tab to set up the “Extra Settings” (the settings depend on each of the Telecom Company). For example, the settings provided by a Telecom Company in Taiwan is

+CGDCONT=1,”IP”,”INTERNET”

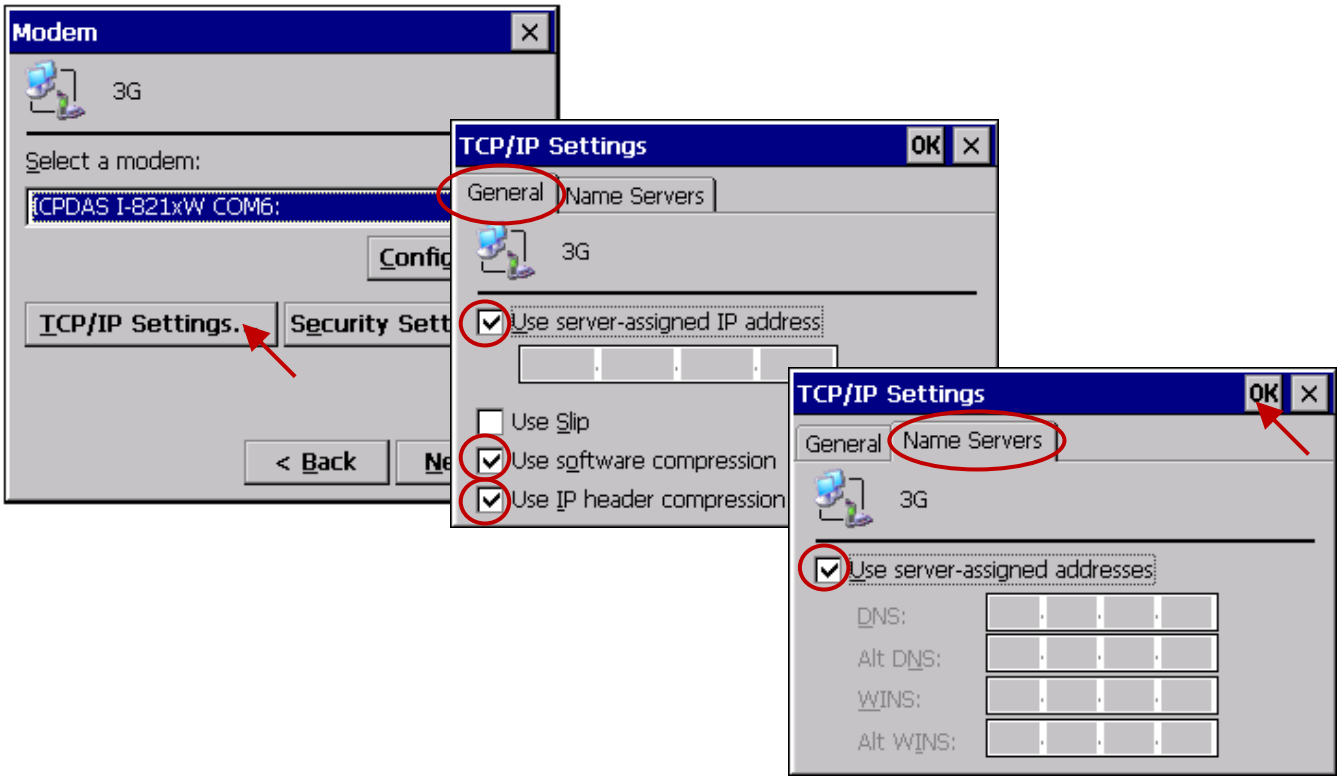
and a Telecom Company in China is

+CGDCONT=1,”IP”,”CMNET”

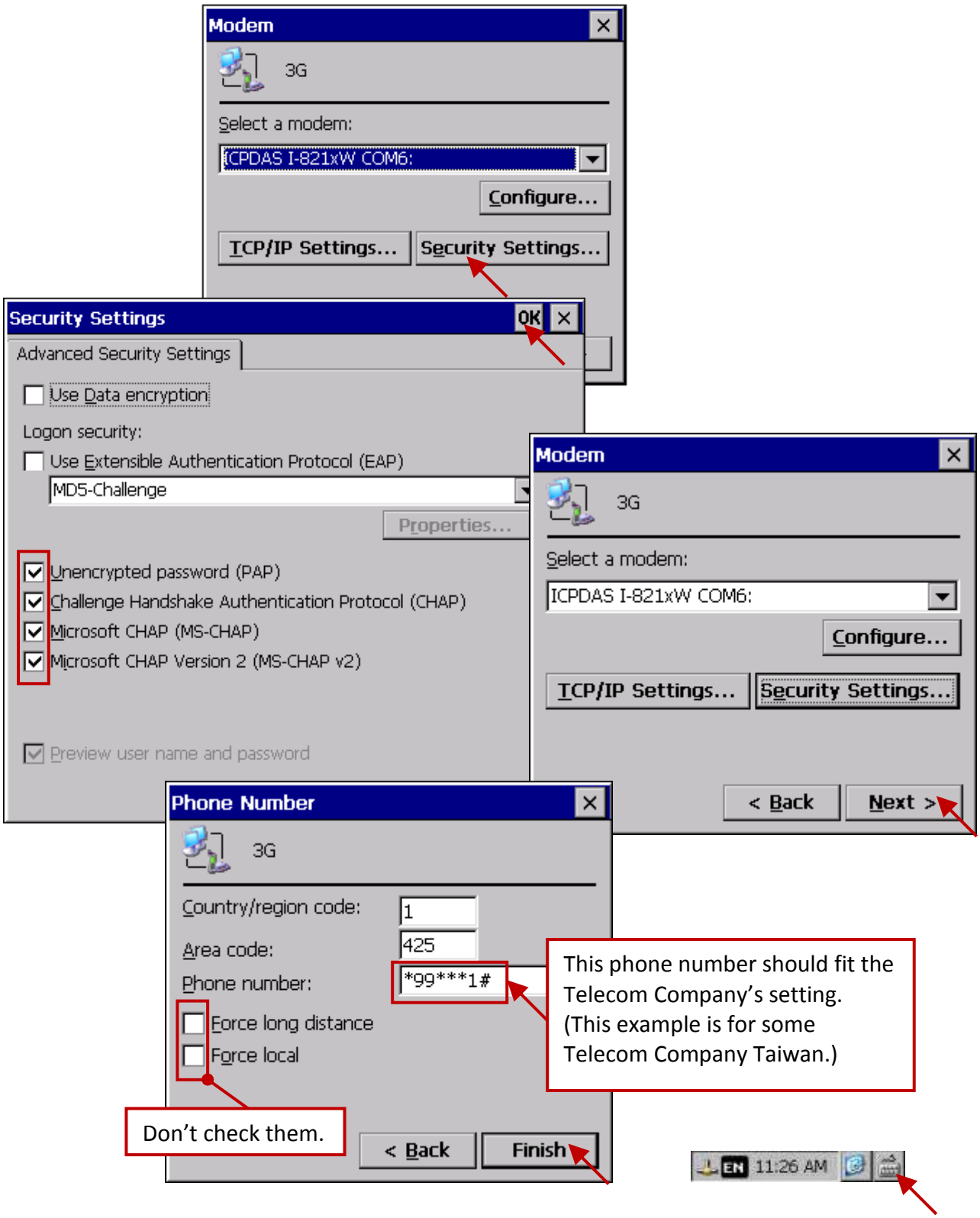
This configuration includes the 3G/2G APN (Access Point Name), please contact your SIM card provider (Telecom Company), to get the settings, or you can also visit the web to search the word “3G/2G APN” to find the settings.



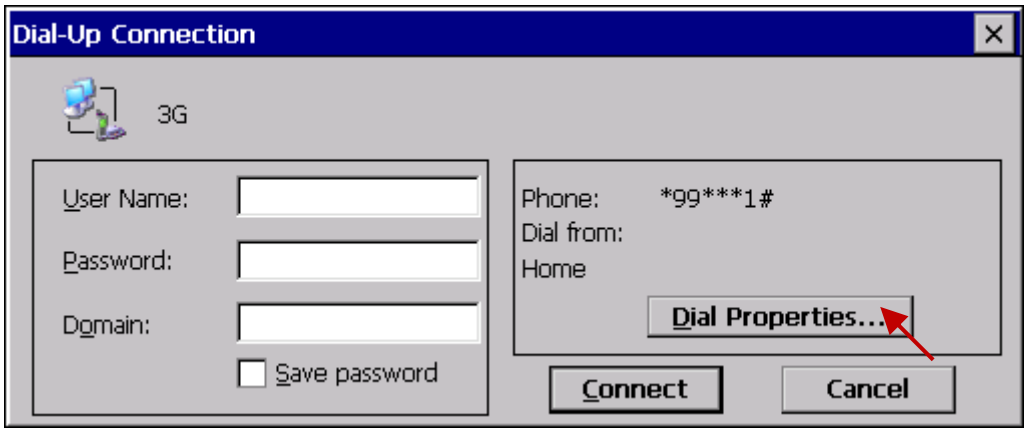
Then get into the “TCP/IP Settings ...” dialog box and follow the same settings as below.

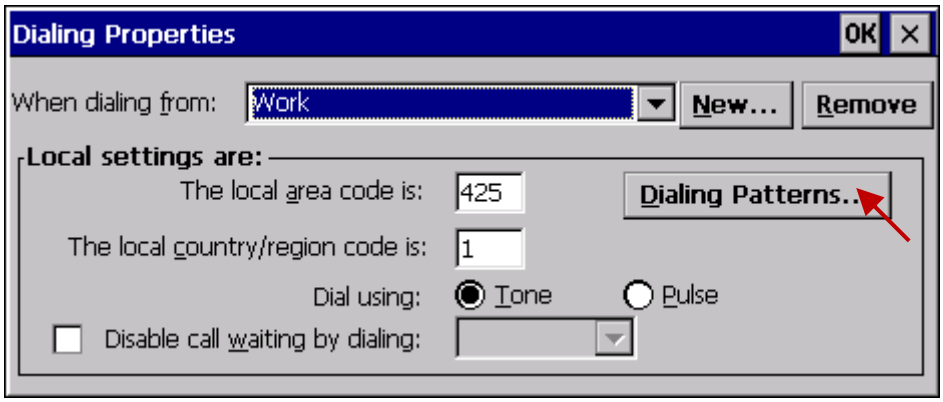


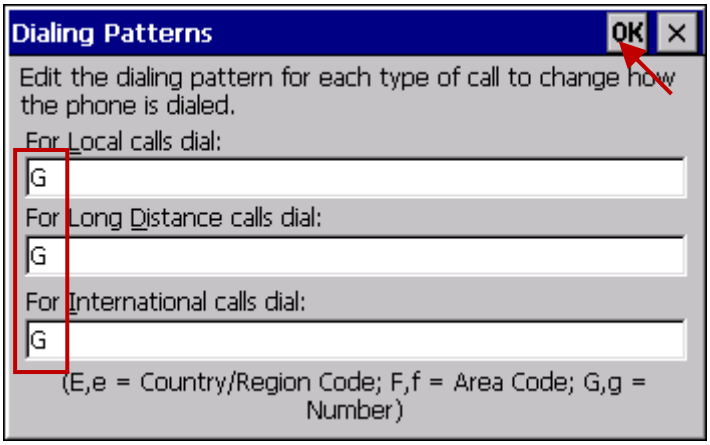
Then get into the “Security Settings” dialog box and follow the same settings as below. Afterward, type the phone number for 3G/2G dial-up, and it must fit for the number provided by Telecom Company, and then click “Finish”.



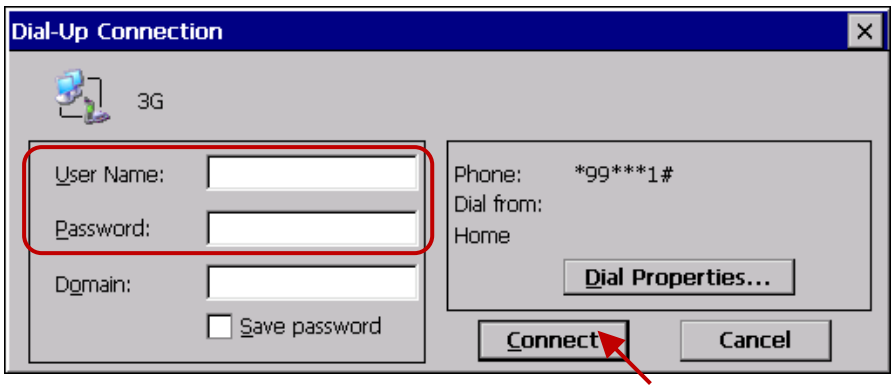
Next, double-click on the new connection (e.g., “3G”) that you have created and get into the “Dial Properties” dialog box, and then get into the “Dialing Patterns” to change the content of those three fields as “G” and click “OK”.



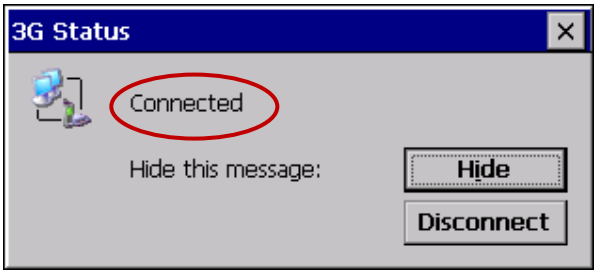




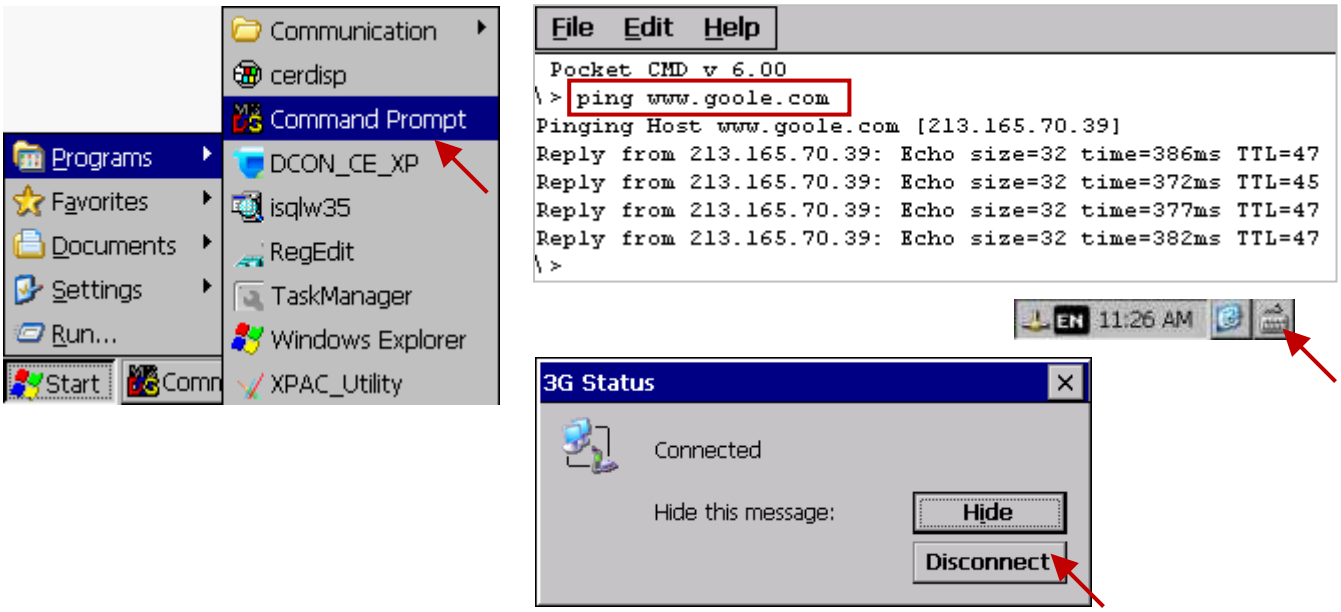
Now, you need to make a dial-up connection to check if the 3G/2G network is OK. Please type the "User Name" and "Password" that provided by the Telecom Company or online search the word "3G/2G APN". As figure below, we use a Taiwan SIM card for Telecom Company as an example (keep two fields blank) and then click "Connect" to make the I-8212W-3GWA (plus SIM card) to start dial-up.



If the connection is successful, it will show up "Connected".
(Note: refer the section 19.2.3 after running the "Ping" command)



After successfully connecting, open "Command Prompt" and give a ping command to check if the connection is fine (If the ping command fails, refer the next section 19.2.3). After a successful ping, it must run "Disconnect", then continue the next important steps.



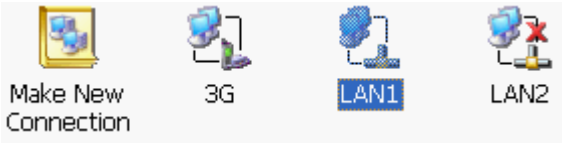
19.2.3 Important Configuration (DO NOT ignore it)!!!

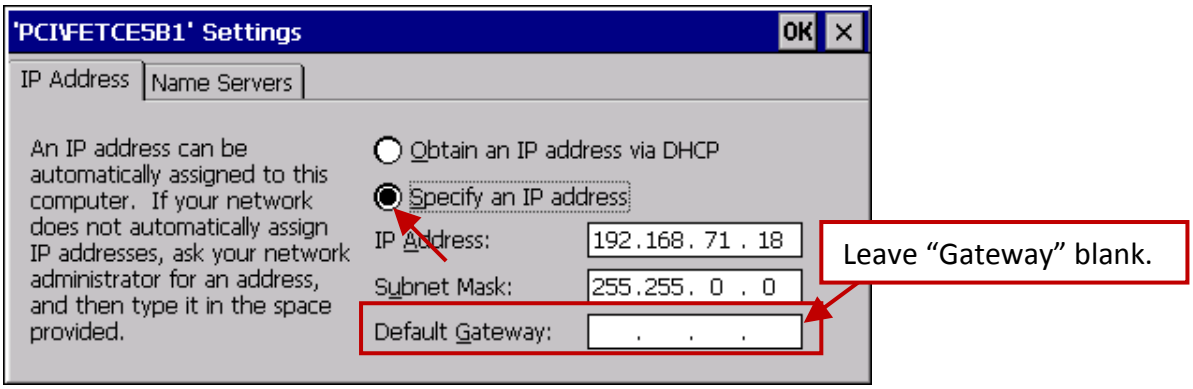
<u>Users must do the following two important settings!</u>

1. If the PAC is going to send/receive the TCP data by using 3G/2G Internet connections, it must clear the gateway settings of LAN1 and LAN2 or else it may not work properly. Then, remember to run the PAC's Utility "File > Save and reboot" once to save the settings.

(Using the XPAC)

(Using the WinPAC)



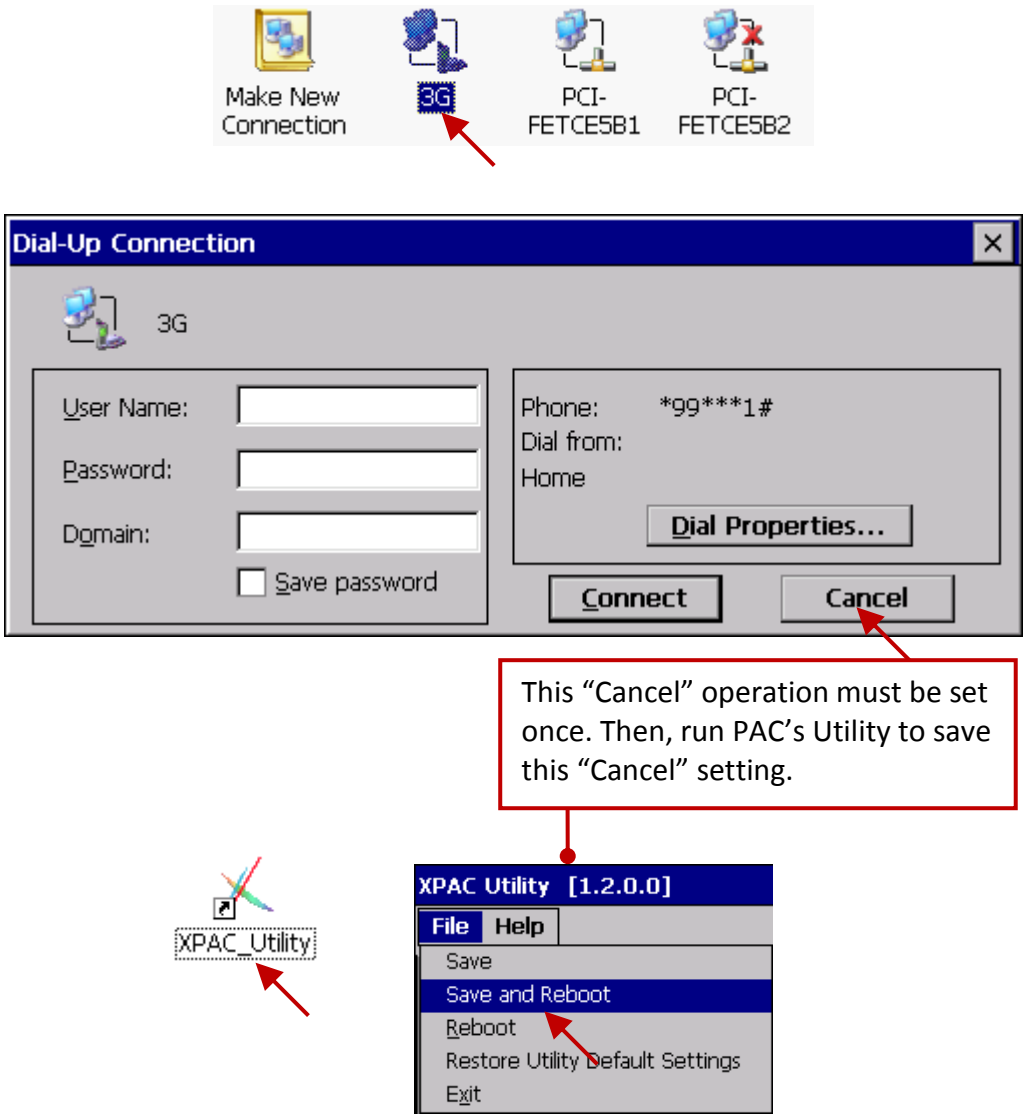


If your PAC can connect to the Internet by using LAN1 or LAN2, recommend not to use the 3G/2G (in such a case, it requires to set the gateway of LAN1 or LAN2). The reason is LAN1 / LAN2 speed is faster than the 3G/2G.

2. In the previous page, the connection has been established. For now, the following configuration is very important and can't be ignored or else it will cause some problem when you connect to the 3G/2G network using the Win-GRAF program.

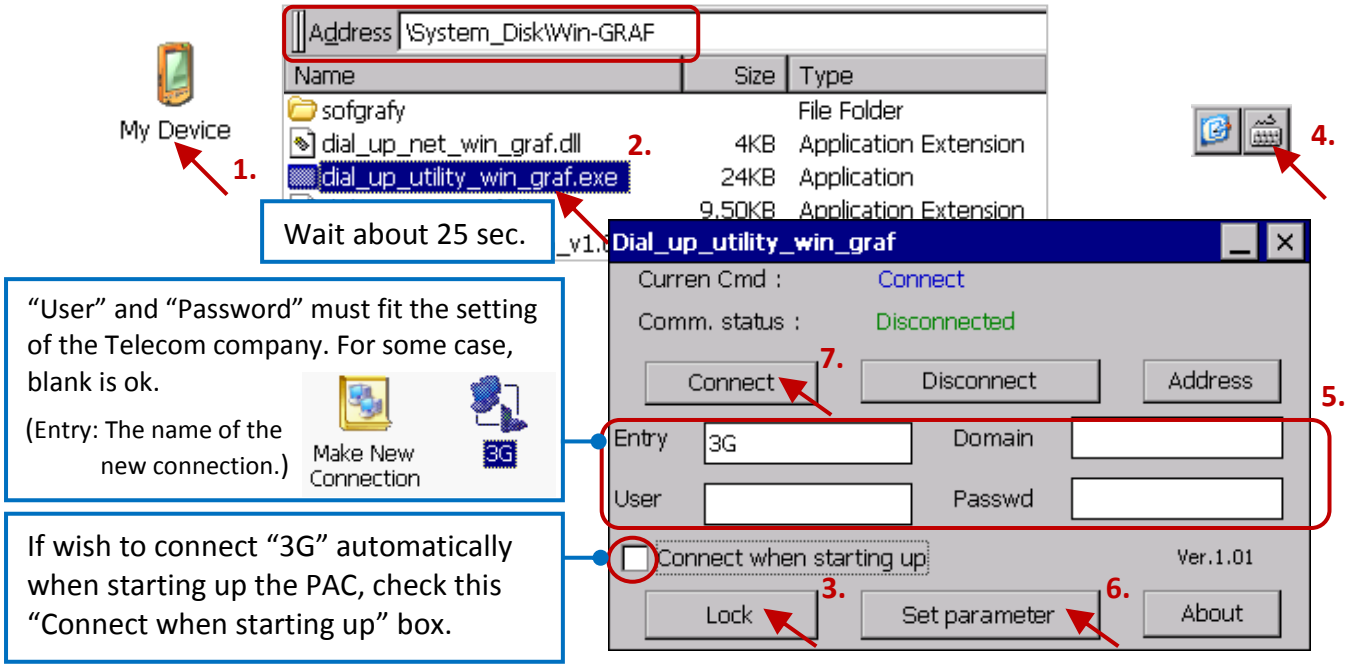
   If the status of 3G/2G connection is still "Connected", please click the "Disconnect" button first.

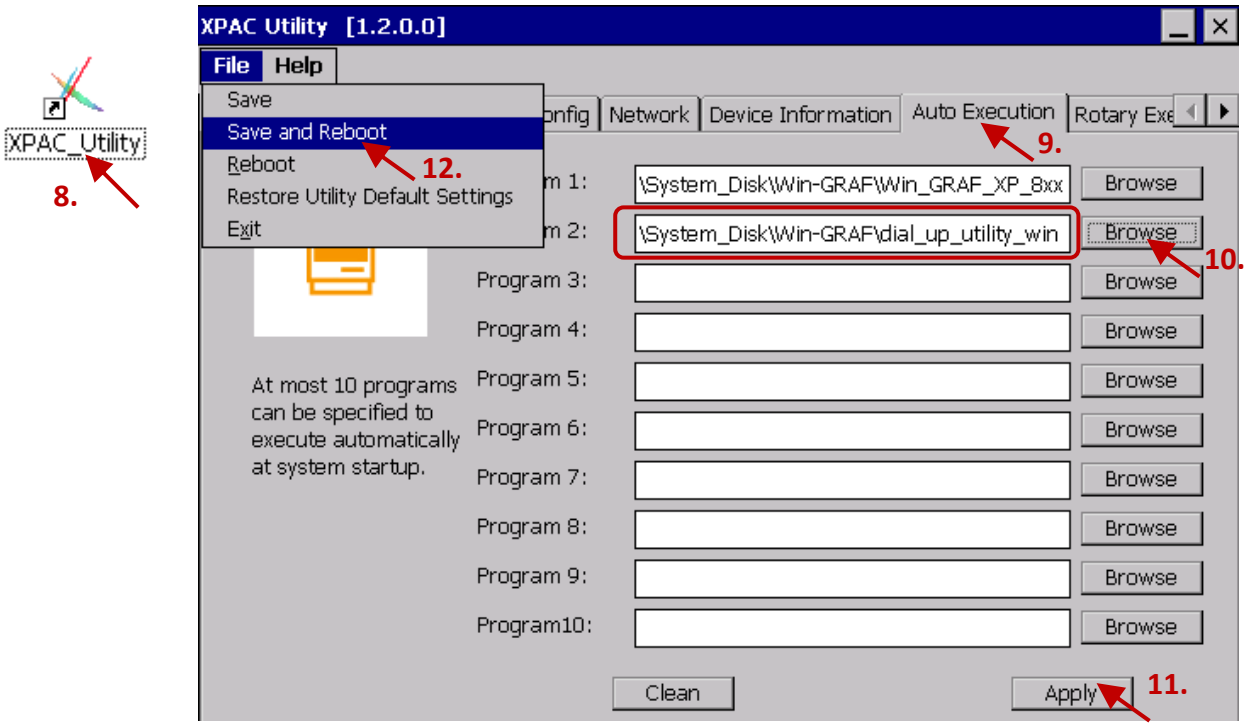
After that, run the new connection (here is “3G”) and then click “Cancel” (At this time, Do Not click “Connect”, **you must click “Cancel”**). Finally, run “File > Save and Reboot” in each PAC Utility (e.g., “XPAC Utility”) to save all the settings (including this and those in the previous section) and then the PAC will restart automatically once.



19.2.4 Enable “Dial_up_utility_win_graf” " Dial-up Software

“Dial_up_utility_win_graf” is a software tool developed by ICP DAS for the 3G/2G dial-up automatically. It allows a Win-GRAF program (or VB.net, C#.net and C program) to connect or disconnect 3G/2G by sending commands and it can also read the connection status or command status. Please follow the steps below to enable this dial-up software. Then, click “Connect” to check if the connection is good and click “Disconnect” to check if the connection is broken. Finally, you need to run “…PAC Utility” and add the “dial_ up_utility_win_graf.exe” to the list of “Auto-Execution” and then run “File > Save and Reboot” to save the settings.





19.3 Function Descriptions for Controlling 3G/2G Connection

This Win-GRAF demo program shows how to use **_3G_connect()** to connect 3G/2G. Set up "To_connect_3G" as TRUE, it will instruct "Dial_up_utility_win_graf" to connect 3G/2G.

```
(* Set "To_connect_3G" as TRUE to connect 3G to
   access to the Internet *)
if  To_connect_3G  then
  To_connect_3G := FALSE ;
  _3G_connect() ;
end_if ;
```

And, this Win-GRAF demo program shows the way to use **_3G_disconnect()** to stop the 3G/2G. Set up "To_disconnect_3G" as TRUE, it will command "Dial_up_utility_win_graf" to disconnect.

```
(* Set "To_disconnect_3G" as TRUE to disconnect 3G *)
if  To_disconnect_3G  then
  To_disconnect_3G := FALSE ;
  _3G_disconnect() ;
end_if ;
```

The program below shows the way to use **_3G_state()** to read the current status of the 3G/2G connection and use **_3G_read_cmd()** to read the current status of the 3G/2G command.

```
(* Get 3G connecting state *)
  State_3G := _3G_state() ;
```

0 : No-action	
(1 ~ 6) : Connecting ...	
1 : Open COM Port	2 : Port opened
3 : Connect device	4 : Device connected
5 : Authenticate	6 : Authenticated
7 : Password expired	8 : Connected
9 : Disconnected	10 : Others

```
(* Read current 3G command
  0 : No action
  1 : Connect
  2 : Disconnect *)
current_3G_cmd := _3G_read_cmd() ;
```

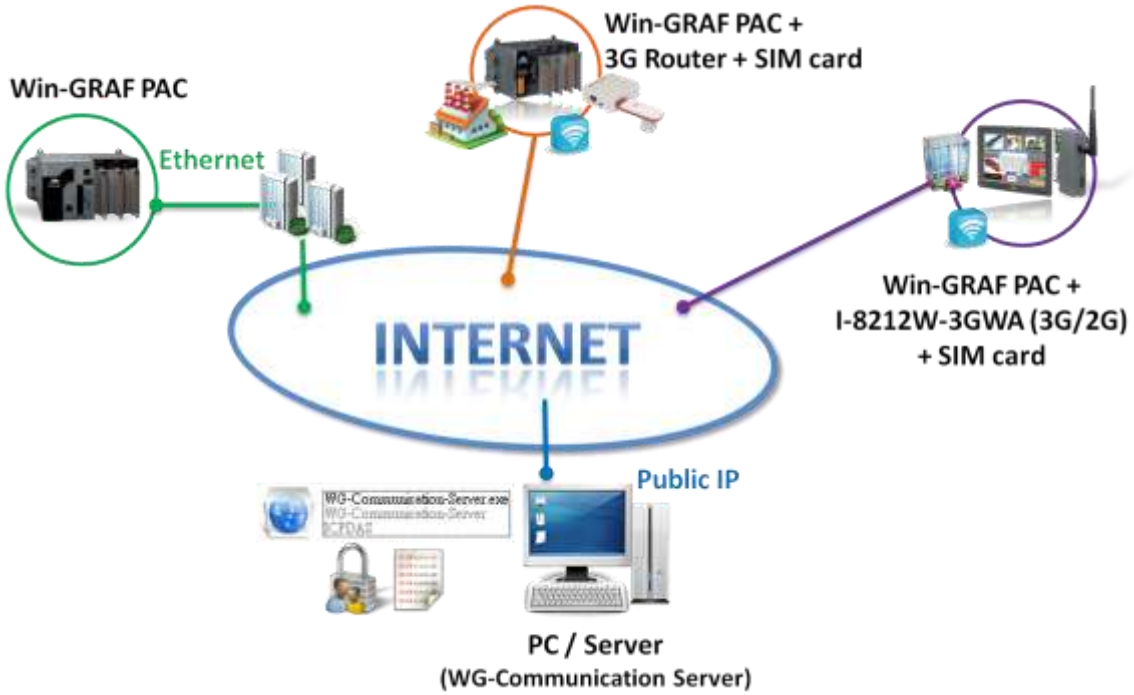
When the 3G/2G dial-up state is "Connected". The driver will try to ping DNS Server and "8.8.8.8" every 15 minutes to test if the 3G/2G communication is ok. If there is no response for 15 seconds, the Win-GRAF PAC will automatically reset the 3G/2G module and then re-dial-up to recover the 3G/2G communication . If the user doesn't want to ping this "8.8.8.8", can modify it to ping one another IP address (for example, 202.43.192.106). Like as below code.

```
(* Set IP address (other than 8.8.8.8) to ping every 15 minutes *)
if  To_ping_a_new_ip  then
  To_ping_a_new_ip := FALSE ;
  TMP_BOOL := _3G_option( 1 , '202.43.192.106' ) ;
end_if ;
```

Chapter 20 Sending a PAC File to a Remote PC via Ethernet or 3G/2G Wireless Networks

For some applications that need to record some useful data on the PAC like temperature, humidity, speed, voltage, current, etc., these data can be saved as a file by the user-designed Win-GRAF program. Then, the user can use the following ways to send this PAC file to a remote PC/Server.

In the shipping CD (CD-ROM: \Napdos\Win-GRAF**demo-project**\), the user can find out the Win-GRAF demo project (demo_send_file.zip) which will be used in this chapter (refer Chapter 12 to restore it). In addition, in the CD-ROM: \Napdos\Win-GRAF**Tools_Utility**\, there is a Win-GRAF Communications Server software (called WG-Communication-Server) which run on a PC/Server to allow the file sending from the PAC.



Note: There are two ways to setup a Win-GRAF PAC with 3G/2G network.
(1) Using a 3G Router plus a SIM card.
(2) Using the I-8212W-3GWA module plus a SIM card (refer Chapter 19).
In addition, the PC/Server must run the "WG-Communication Server" software which must configure a public IP for the user sends the PAC file by using 3G/2G network or Ethernet (WAN).

The following Win-GRAF driver versions (or later version) support functions described in this chapter.

WP-8xx8 :	1.05
VP-x2x8-CE7 :	1.01
XP-8xx8-CE6 :	1.03
WP-5xx8-CE7 :	1.02

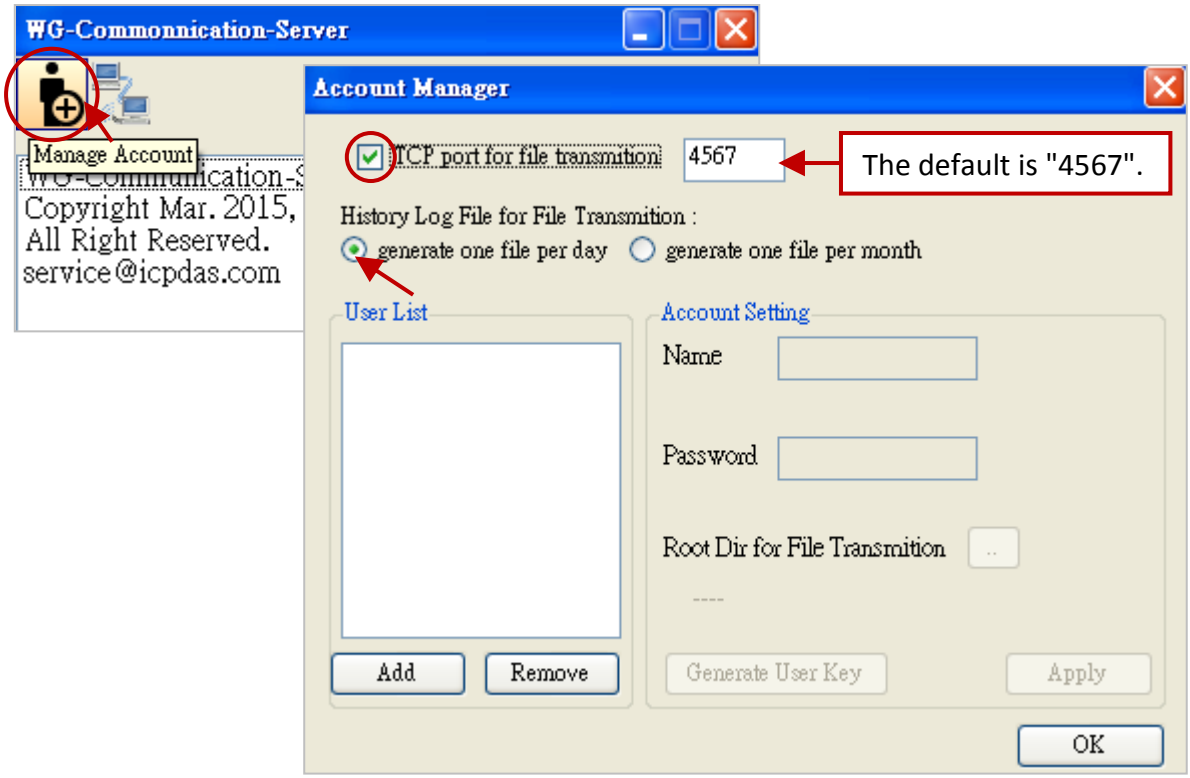
20.1 Description of the "WG-Communication-Server" Software

Note: **Copy the "WG-Communication-Server" folder (includes WG-Communication-Server.exe and DLL) into the D:\ on your PC/Server, and then running this software to allow the file sending from a remote PAC.** The user can use the "WG-Communication-Server" to create the username/ password (max. 100) for a remote PAC can log in and send a file to this PC/Server.

Add a User Account:

1. After running the "WG-Communication-Server", it will zoom out to the bottom-right corner of your desktop screen (running in the background). Double-click the small icon if you want to configure it.

2. Click "Manage Account" to open Account Manager and then check "TCP port for file transmission".



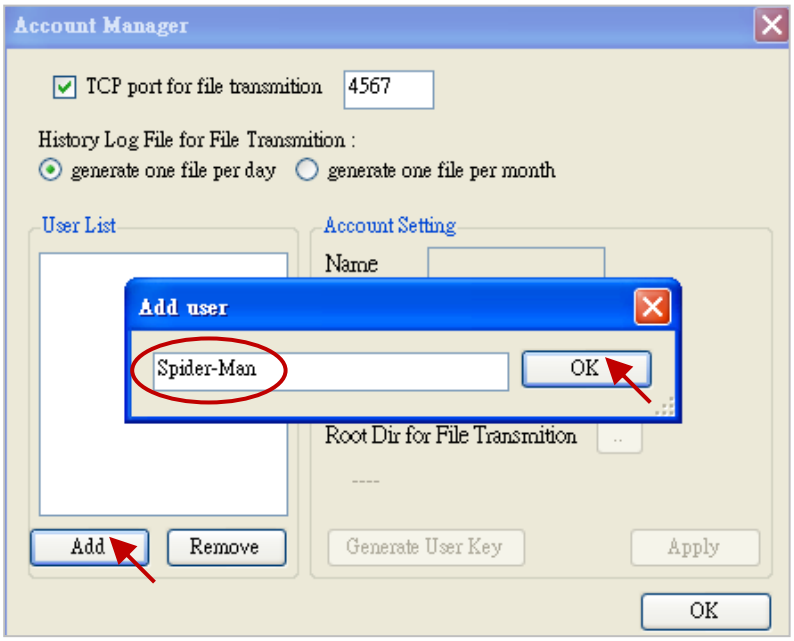
"TCP port for file transmission": To enable the specific TCP port (Default: 4567; Range: 1000 ~ 9999) for communicating with the PAC while file sending. (**Note:** Whether check or uncheck this item, or even modify the TCP Port number, the user must restart this software to apply the setting.)

"History Log File":

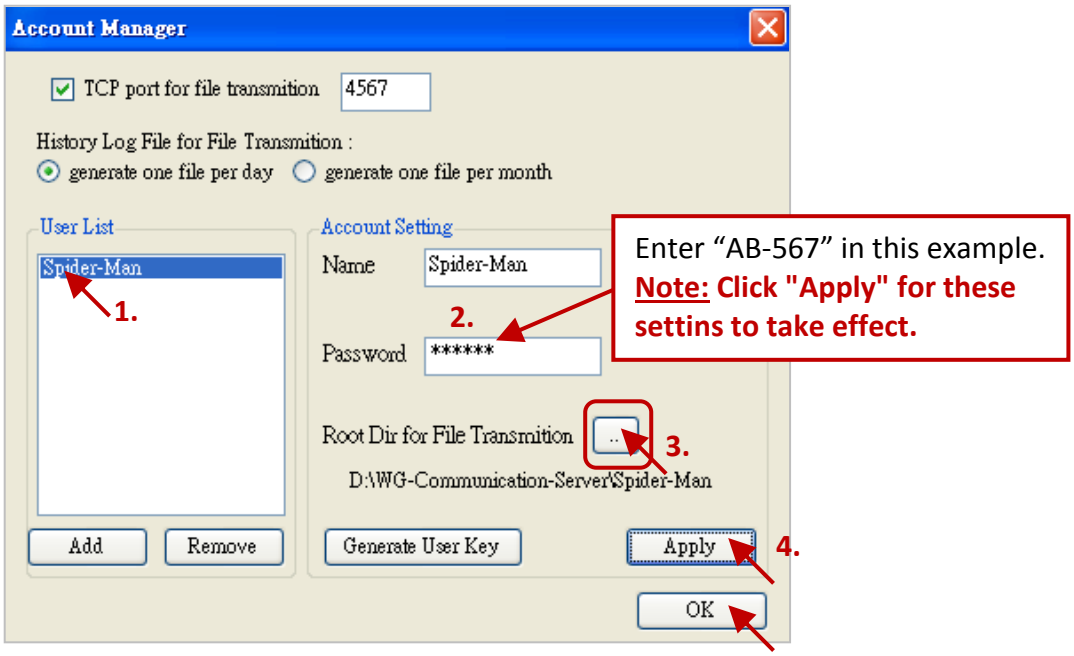
To generate one historical log file per day/month (choose "per day" in this case).

Note: Go to the next step to set up an account. In addition, the user can open the "D:\WG-Communication-Server\account.txt" to check the account that you set up before.

3. Click the "Add" button to enter a username (e.g., "Spider-Man") and then click "OK".

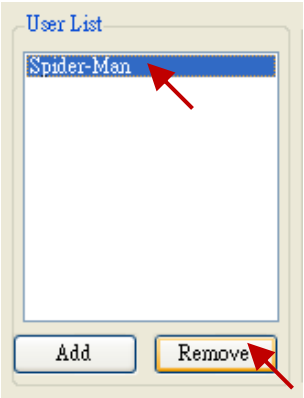


4. Click on the username (e.g., "Spider-Man") to set its password (e.g., "AB-567"). Click "Root Dir" can set the PAC file storage path, we recommend to use the defaults - D:\WG-Communication-Server \User Name (e.g., Spider-Man), and then click "Apply" to take effect. Finally, click "OK".
   Note: The user can view the "Generate User Key" usage in Section 22.1.



Delete a User Account

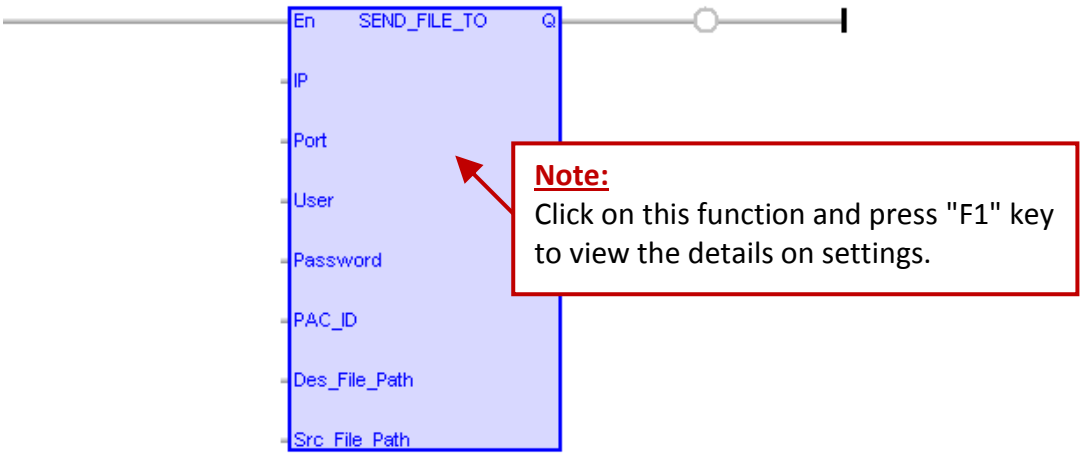
Click on the username you want to delete (e.g., "Spider-Man"), click the "Remove" button, and click "OK" to delete this account (username/password).



20.2 "Send_File_To", "Send_File_State" and "Send_File_Abort" Functions

There are 3 functions which can handle the file sending from the PAC to the PC.

The "Send_File_To" function:
To send one PAC file to a remote PC/Server which is running the "WG-Communication-Server".



Input parameters:

Note: **Valid characters for the "User", "Password", "Des_File_Path" and "Src_File_Path":**
A ~ Z, **a ~ z** (case-sensitive), **0 ~ 9**, '**.**' (dot), '**@**' (At), '**-**' (minus) and '**_**' (underscore).

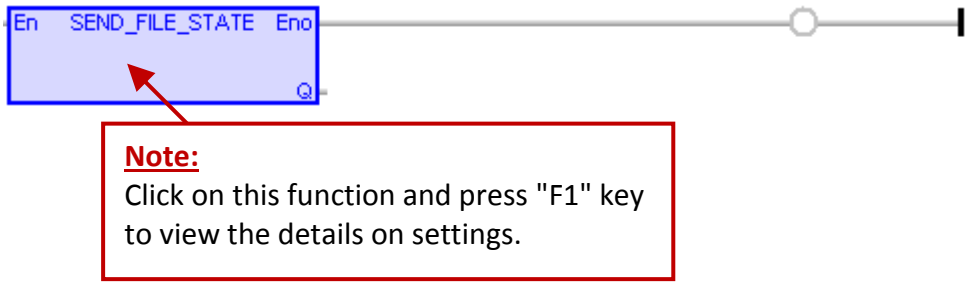
IP: (Data type: String) IP address of the remote PC (e.g., `192.168.71.29').
Port: (Data type: DINT) The TCP port number (range: 1000 ~ 9999).
User: (Data type: String) User name (max. 32 characters).
Password: (Data type: String) Password (max. 32 characters).
PAC_ID: (Data type: DINT) A number to identify the file is sent by which PAC.
Value range: -2,147,483,648 ~ 0 ~ 2,147,483,647.
Des_File_Path: (Data type: String) Destination file path in PC (max. 128 characters).
And the first character should be '****', however the last character cannot be '\'.
(E.g. '****2014\12\data001.txt' or '****Record\recp-2014-11-08.txt')
Src_File_Path: (Data type: String) Source file path in PAC (max. 128 characters).
And the first character should be '****', however the last character cannot be '\'.
(E.g. '****Micro_SD\PAC\data001.txt' or '****System_Disk\DATA\recp-2014-11-08.txt')

Output parameters:

Q: (Data type: BOOL)
TRUE: Communication OK.
FALSE: Wrong input parameters or "Src_File_path" doesn't exist or file size is 0.

The "Send_File_State" Function:

To get the sending state of the PAC file. (MUST use with the "Send_File_to" function.)



Input parameters:

None.

Output parameters:

Q: (Data type: DINT)

0 : Sleep, no "Send_File_To( )" function is called.

1 ~ 99 : Busy, a file is sending now. ("1 ~ 99" means the percentage of completion)

100 : Succeed, the file sending is finished and the file is sent successfully.

-1 : Send failed or timeout.

-2 : The file sending is interrupted by the "Send_File_Abort" command.

-3 : Username/Password error.

-4 : Unable to create a sub-folder or file in the PC, or
The file is over 10,000,000 Bytes, or
There is no "WG-Communication-Server" running on the PC.

-5 : The file for sending does not exist or the file size is "0".

-6 : Unable to open the file located at "\Email_ETH" of the PAC.

The "Send_File_Abort" Function:

To abort the file sending.

Input parameters:

None.

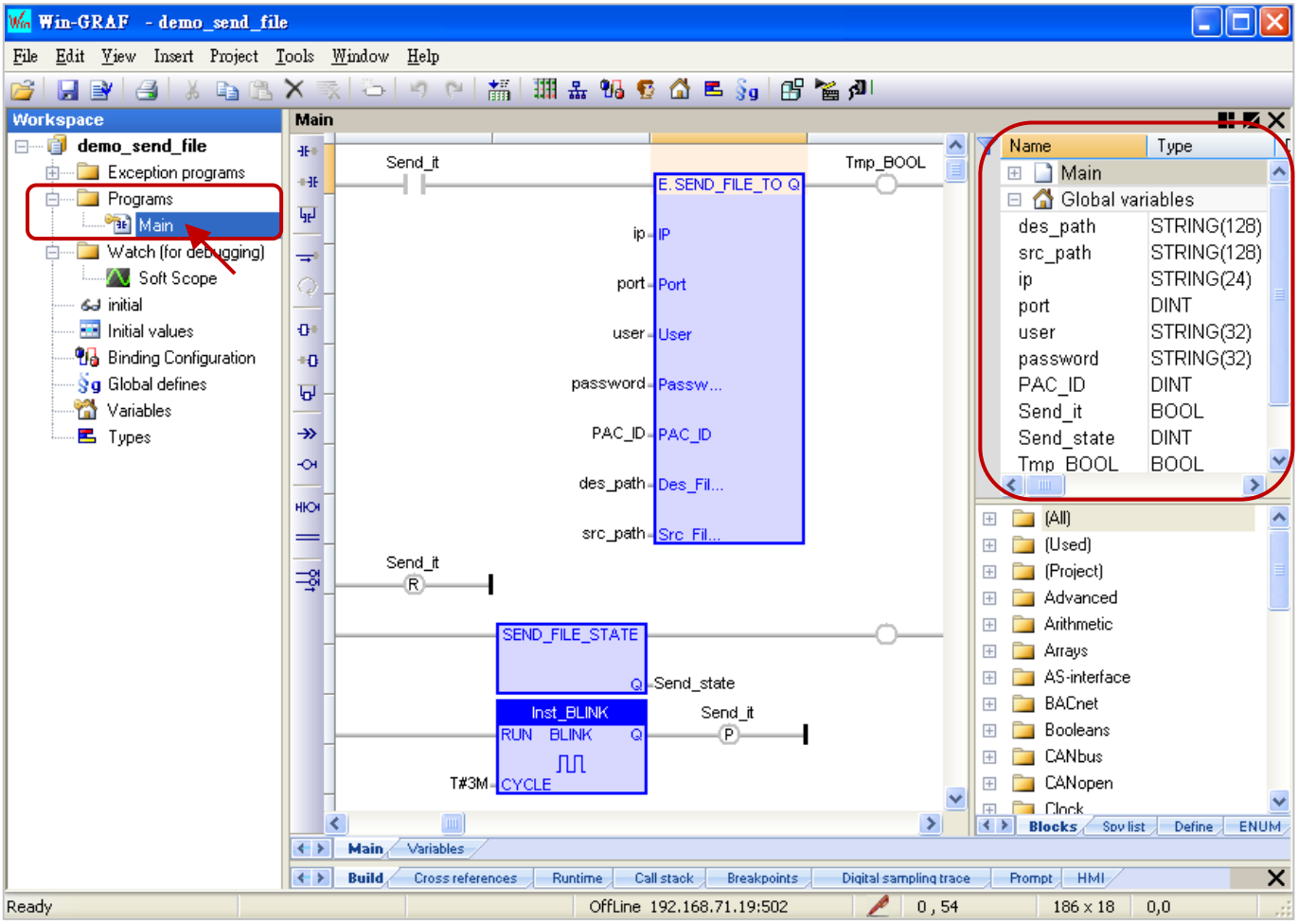
Output parameters:

Q: (Data type: DINT), always return "TRUE".

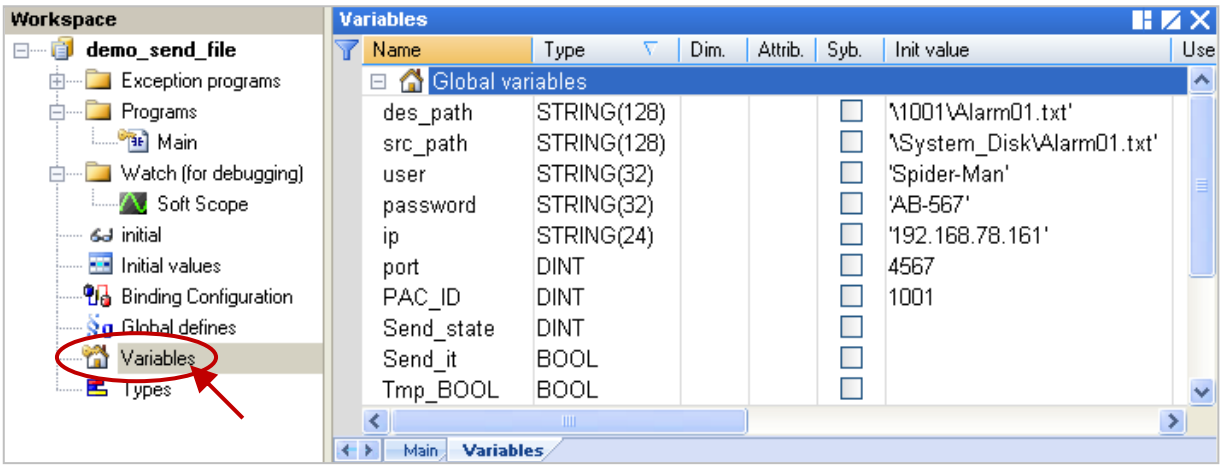
20.3 Description of the Win-GRAF Demo Project (demo_send_file.zip)

This Win-GRAF demo project (demo_send_file.zip) can be found in the shipping CD (CD-ROM: \Napdos\Win-GRAF\demo-project\), refer Chapter 12 to restore it.

After opening this "demo_send_file" project, mouse double-click on "Main" to open this LD program. Then, you can see all defined variables listed on the right of the window (i.e., Variable Area).



First of all, we need to know what variables are used in this project and their purposes. You can also click "Variables" on the left of the window (i.e., Workspace) to view/set all variables.



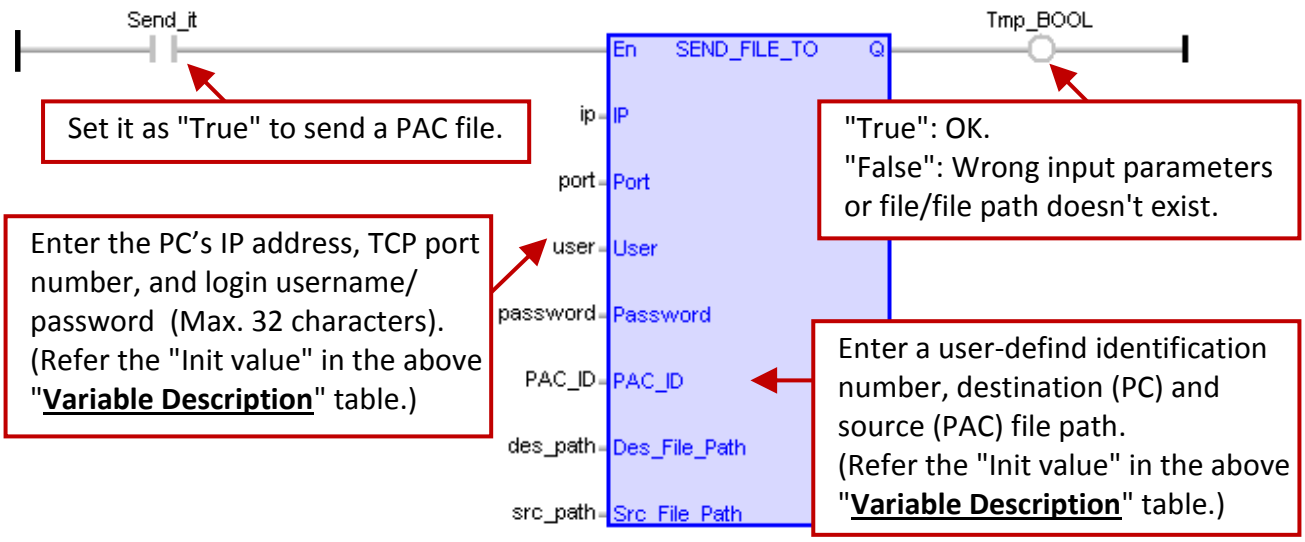
Variable Description:

The following table describes the variable name, data type, their usages, and initial value that used in the Win-GRAF project (demo_send_file.zip).

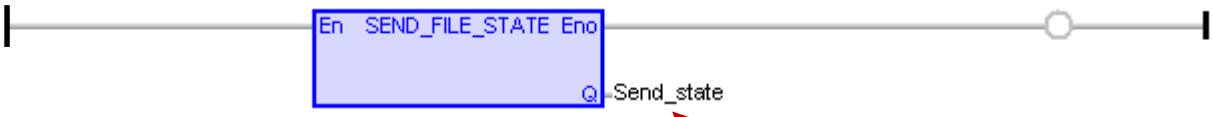
Name	Data Type	Description
des_path	STRING(128)	Destination file path in PC (max. 128 characters). (Init value: **'\1001\Alarm01.txt'**)
src_path	STRING(128)	Source file path in PAC (max. 128 characters). (Init value: **'\System_Disk\Alarm01.txt'**)
user	STRING(32)	PC login username (Max. 32 characters). (Init value: **'Spider-Man'**)
password	STRING(32)	PC login password (Max. 32 characters). (Init value: **'AB-567'**)
ip	STRING(24)	IP address of the remote PC. (Max. 24 characters.) (Init value: **'192.168.78.161'**)
port	DINT	The TCP port number of the WG-Communication-Server to receive files. (Init value: **1000** ; Range: 1000 ~ 9999).
PAC_ID	DINT	A number to identify the file is sent by which PAC. (Init value: **1001**)
Send_state	DINT	Get the sending state of the PAC file. (See Section 20.2)
Send_it	BOOL	Set it as "True" to send a PAC file.
Tmp_BOOL	BOOL	TRUE: Communication OK. FALSE: Wrong input parameters or "Src_File_path" doesn't exist or file size is 0.

LD Program (Main)

The "Send_File_To" function (see Section 20.2) is used to send a PAC file to a remote PC/Server.



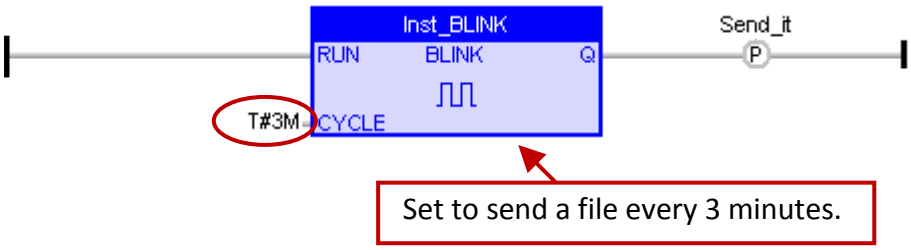
The "Send_File_State" function is used to get the sending state of the PAC file.



Return the state of file sending.

0 :	Sleep, no "Send_File_To( )" function is called.
1 ~ 99 :	Busy, a file is sending now. ("1 ~ 99" means the percentage of completion)
100 :	Succeed, the file sending is finished and the file is sent successfully.
-1 :	Send failed or timeout.
-2 :	The file sending is interrupted by the "Send_File_Abort" command. (see Section 20.2)
-3 :	Username/Password error.
-4 :	Unable to create a sub-folder or file in the PC, or The file is over 10,000,000 Bytes, or There is no "WG-Communication-Server” running on the PC.
-5 :	The file for sending does not exist or the file size is "0".
-6 :	Unable to open the file located at “\Email_ETH” of the PAC.

The "BLINK" function block in this example is used to send a file every 3 minutes.



20.4 Test for File Sending

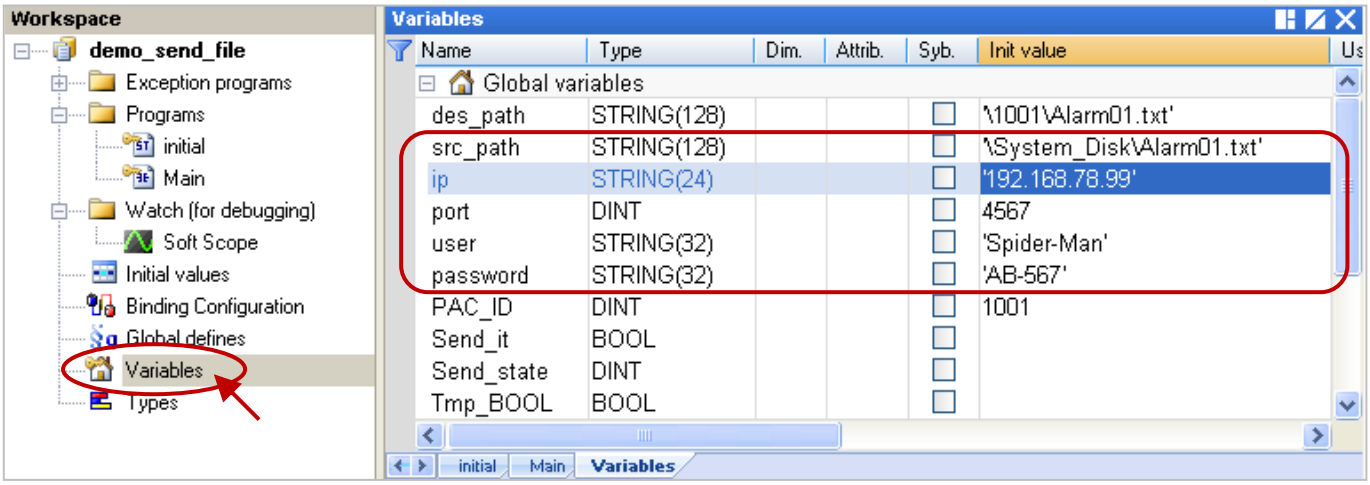
Before testing this project, check the following settings in the "Variables" window:

"src_path": Does the file exist on the PAC (i.e., \System_Disk\Alarm01.txt)? If not, download it by using FTP or assign other file. (**Note:** the file size cannot be 0 bytes.)

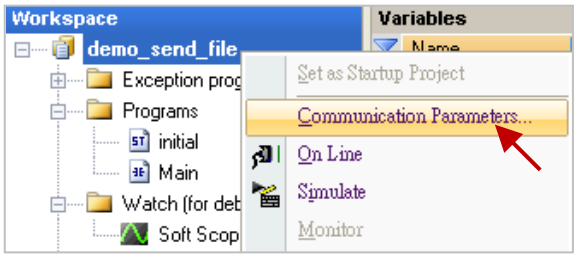
"ip": Enter the IP address of your PC. (If using 3G/2G network or an Ethernet (WAN) to connect to a Server, the user must enter a public IP.)

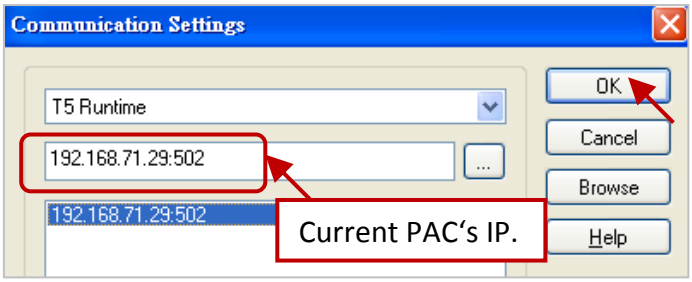
"port" (4567)**,"user" and "password":**

These settings must be the same as the "WG-Communication-Server" settings (See Section 20.1).



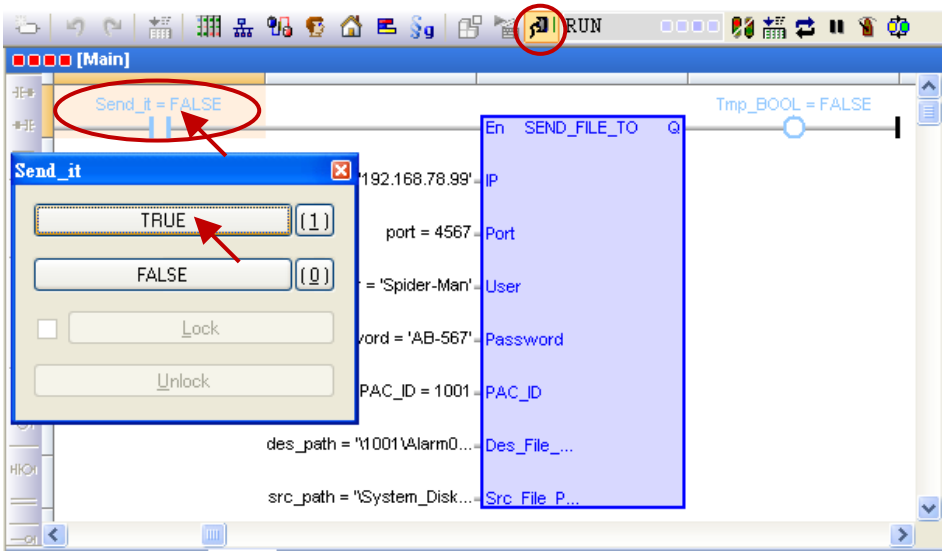
Moreover, set the current IP address of the PAC in the Communication Settings, and then recompile and download this project to the Win-GRAF PAC.





Note: If the user wants to set the timeout value (default: 3 seconds), see Section 2.3.5.
(E.g. Set the IP to "192.168.71.29:502**(10)**" which means the timeout is **10** seconds.)

After downloading the project, double-click "Send_it" in the LD program (i.e., Main) to set it as "TRUE" to start sending the file.

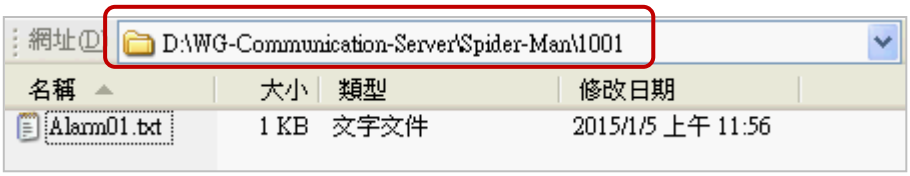


When the progress of "Send_state" reaches "100", it means that the file is sent successfully.

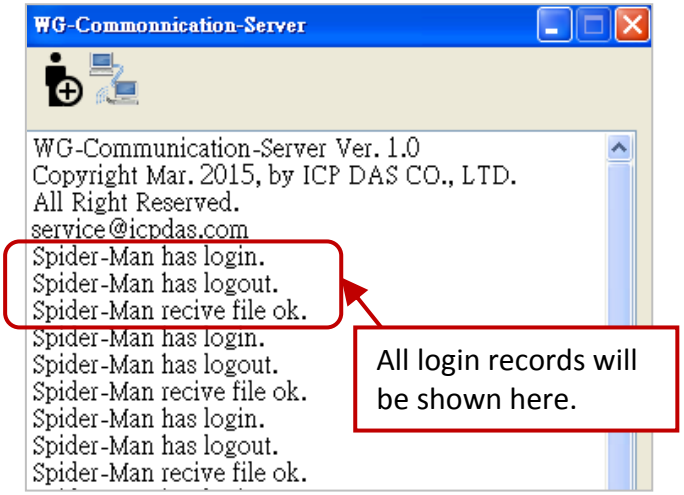
Return the state of file sending.

0 :	Sleep, no "Send_File_To( )" function is called.
1 ~ 99 :	Busy, a file is sending now. ("1 ~ 99" means the percentage of completion)
100 :	**Succeed, the file sending is finished and the file is sent successfully.**
-1 :	Send failed or timeout.
-2 :	The file sending is interrupted by the "Send_File_Abort" command. (see Section 20.2)
-3 :	Username/Password error.
-4 :	Unable to create a sub-folder or file in the PC, or The file is over 10,000,000 Bytes, or There is no "WG-Communication-Server" running on the PC.
-5 :	The file for sending does not exist or the file size is "0".
-6 :	Unable to open the file located at "\Email_ETH" of the PAC.

Now, the file is sent to the PC - "D:\WG-Communication-Server\Spider-Man\1001\Alarm01.txt". (Refer Section 20.1 – Step4 and Section 20.3 – the "des_path" variable)



In addition, this Win-GRAF example project (demo_send_file.zip) is designed to send a file every 3 minutes, the user can open "WG-Communication-Server" to view file receving records.



Chapter 21 Win-GRAF SMS Function

This chapter shows the way to send/receive a text message by using the Win-GRAF PAC comes with the 3G/2G wireless module. This Win-GRAF demo project (Demo_SMS.zip) can be found in the shipping CD (CD-ROM: \Napdos\Win-GRAF\demo-project\), refer the Section 21.2 for details).

Software/Hardware Requirements:

1. The Win-GRAF PAC
   The following Win-GRAF driver versions (or later) support the SMS function (refer Chapter 19).
   WP-8xx8 : 1.05 ; VP-x2x8-CE7 : 1.01 ; XP-8xx8-CE6 : 1.03 ; WP-5xx8-CE7 : 1.02
2. The 3G/2G Wireless Module
   Visit the website http://m2m.icpdas.com/2G_3G_Modems.html for more details.
   All supported GSM Modems: GTM-201-RS232, GTM-203M-3GWA, I-8212W, I-8212W-3GWA (see Chapter 19), I-8213W-3GWA.
   Note: Due to the product certification issue, these modules can sale in certain areas.
   Please contact our agents for more information.

21.1 "GSM_Open", "Send_SMS" and "Read_SMS" Functions

There are 3 functions which can handle the SMS messaging by the Win-GRAF PAC.

The "GSM_OPEN" function:

To open/close the GSM module.

Note:
Click on this function and press "F1" key to view the details on settings.

Input parameters:

gEnable: Data type: BOOL
TRUE: Open the specified PAC's COM port to connect the GSM module and initialize it.
FALSE: Disconnect the GSM module and close the specified PAC's COM port.

gCOMPort: Data type: DINT
The PAC's COM port number which connects with the GSM module.

gPinCode: Data type: STRING
Using this PIN code to unlock the SIM card, if it is necessary.

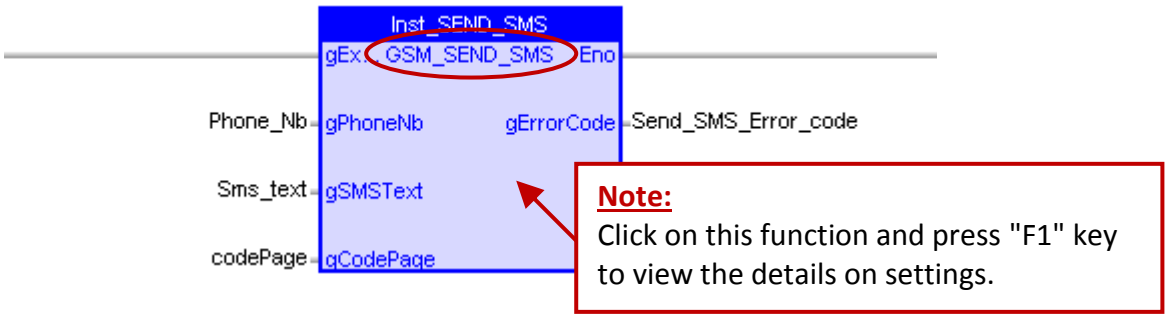
Output parameters:

gIsInitialized: Data type: BOOL
TRUE: Open the specific PAC's COM port and initialize the GSM module successfully.
FALSE: Failed to open the specific PAC's COM port or the GSM module is not initialized.

gSignalQuality: Data type: SINT
0 ~ 31: The higher value means that the signal strength is stronger.
99: Unknow or not detectable.

gErrorCode: Data type: INT
0 : No error.
-1 : Broken line.
-2 : SIM card not inserted.
-3 : SIM PIN code wrong.
-4 : SIM configuration error.
-5: Cannot open the specified PAC's COM port.

The "GSM_SEND_SMS" Function:
To send the SMS message via the GSM module.



Notice: Before using this "GSM_SEND_SMS" function, first the user must use the "GSM_OPEN" to open the PAC's COM Port which the GSM module connects with, or else this function will not work.

Input parameters:

gExecute: Data type: BOOL
Pulse TRUE: trigger it to send SMS message.

gPhoneNb: Data type: STRING
Destination-Address (i.e., the phone number).

gSMSText: Data type: STRING
The text message.

gCodePage: Data type: UDINT
The code page of the text.
English: 0 Traditional Chinese: 950 Simplified Chinese: 936
Japanese: 932 Russian: 866

Notice: If the "gCodePage" is set as "0", the max length of text is 160 characters.
If the "gCodePage" is not set as "0", the max length of text is 70 characters.
If the user type more than the maximum text length, the Win-GRAF driver will just send the maximum characters from the beginning of the text.

Output parameters:

gErrorCode: Data type: INT

4 : Send SMS succeeds	-1 : Cannot find valid GSM module.
3 : Sending SMS	-2 : SIM card not inserted.
2 : Send SMS is pending	-4 : SIM card configuration failed.
1 : Prepare to send SMS	-5 : Cannot open the PAC's COM port.
0 : No operation.	-6 : NO recipient number.
	-7 : Send SMS message failed.

The "GSM_READ_SMS" Function:

To read the SMS message via the GSM module.

Note:
Click on this function and press "F1" key to view the details on settings.

Notice: Before using this "GSM_READ_SMS" function, first the user must use the "GSM_OPEN" to open the PAC's COM Port which the GSM module connects with, or else this function will not work.

Input parameters:

gExecute: Data type: BOOL
TRUE : Enable to read the SMS message from the GSM module.
FALSE : Disable to read the SMS message from the GSM module.

gCodePage: Data type: UDINT
The code page of the text. Please refer the following example:
English: 0 Traditional Chinese: 950 Simplified Chinese: 936
Japanese: 932 Russian: 866

Output parameters:

gIsNewSMS: Data type: BOOL
Pulse TRUE: the new message is coming.

gSenderPhoneNb: Data type: STRING
Originating-Address (i.e., the phone number).

gReceiveSMSText: Data type: STRING
The text message.

21.2 Description of the Win-GRAF Demo Project (Demo_SMS.zip)

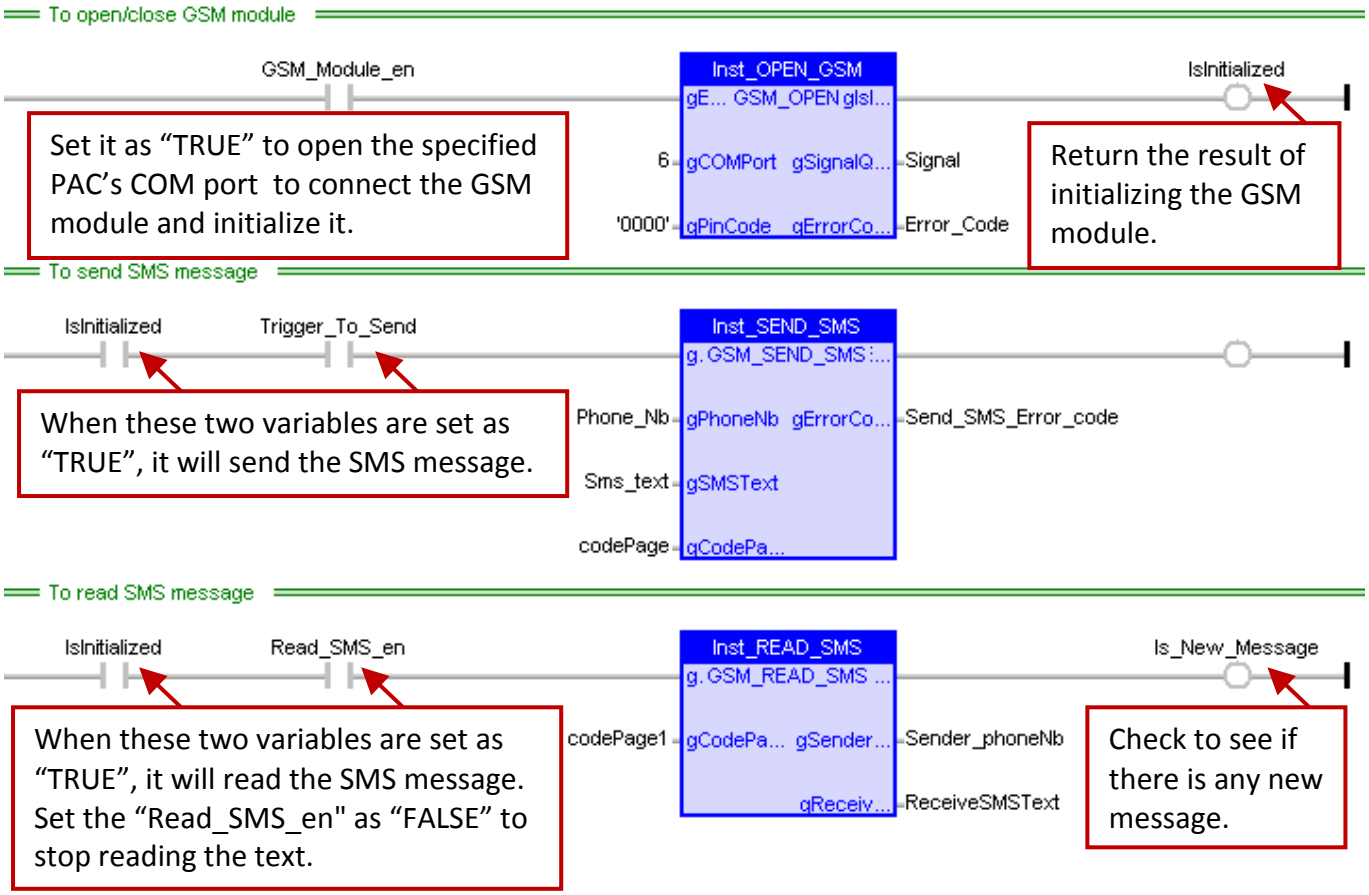
This Win-GRAF demo project (Demo_SMS.zip) can be found in the shipping CD (CD-ROM: \Napdos \Win-GRAF\demo-project\), refer Chapter 12 to open project from a zip file. There are one LD program (SMS_pro) and two ST programs (SendMessage and ReadMessage) in this "Demo_SMS" project.

Variable Description: (* : Refer the Section 21.1 for more details)

Name	Data Type	Description
GSM_Module_en	BOOL	Set it as “TRUE” to open the specified PAC’s COM port to connect the GSM module and initialize it.
Signal	SINT	The signal quality of the GSM module. (*)
Error_Code	INT	The error code of the GSM module. (*)
IsInitialized	BOOL	To check if the GSM module has been initialized.
Trigger_To_Send	BOOL	Set it as “TRUE” to send a text message.
Phone_Nb	STRING(255)	The phone number of recipient. (Init value: '0932860424')
Sms_text	STRING(255)	The content of texting. (Init value: 'This message is sent from Win-GRAF PAC')
codePage	UDINT	The code page of the sending text. (*)
Send_SMS_Error_code	INT	The error code during the SMS sending. (*)
Read_SMS_en	BOOL	Set it as “TRUE” to read a text message.
codePage1	UDINT	The code page of the receiving text. (*) (Init value: UDINT#950)
Sender_phoneNb	STRING(255)	The phone number of sender.
ReceiveSMSText	STRING(255)	The received text message.
Is_New_Message	BOOL	To check if there is any new text message.
Got_New_Message	STRING(255)	To receive text message. (ST – ReadMessage)
Got_Message_from_who	STRING(255)	To receive the phone number of sender. (ST – ReadMessage)

The LD Program (SMS_pro)

Refer the Section 21.1 to view how to configure these three functions.



The ST Program (SendMessage)

```
if Trigger_To_Send then

  if IsInitialized then
    if Send_SMS_Error_code < 0 then
      (* Send SMS failed *)
      Trigger_To_Send := false;

      (* TODO: Add failed processing here *)

    elsif Send_SMS_Error_code = 4 then
      (* Send SMS successed *)
      Trigger_To_Send := false;

      (* TODO: Add success processing here *)

    end_if;
  else
    (* GSM module is not initialized *)
    (* TODO: Add failed processing here *)

  end_if;
end_if;
```

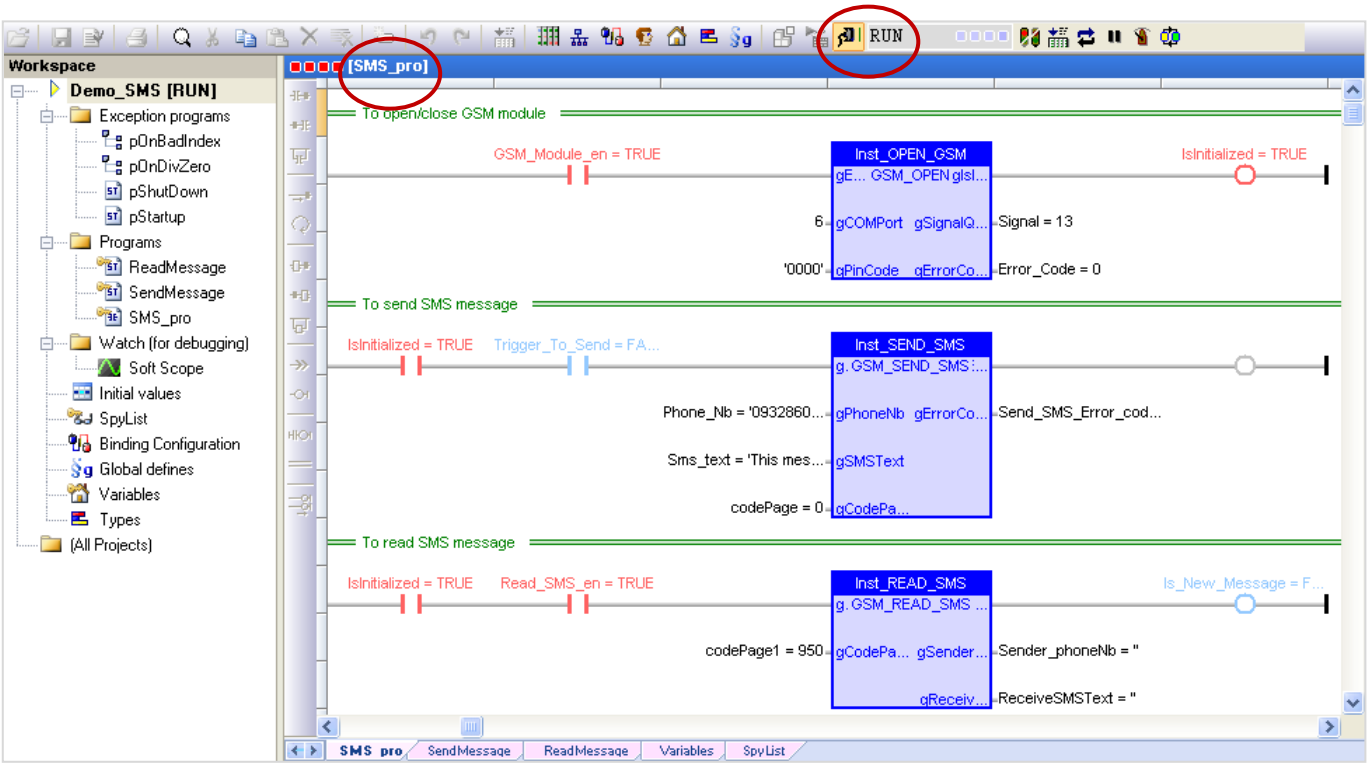
The ST Program (ReadMessage)

```
(* Get new message *)
if Is_New_Message then

   (* If the user need more time to deal with the new message, *)
   (* the user could set "Read_SMS_en" as “FALSE” to disable the Read_SMS function *)
   (* and then set it as “TRUE” after dealing *)
   (* Notice : If the user set "Read_SMS_en" as “FALSE”, it will stop reading the SMS message. *)
   (* Read_SMS_en := false; *)

   Got_New_Message := ReceiveSMSText;
   Got_Message_from_who := Sender_phoneNb;

   (* do more operating *)

end_if;
```

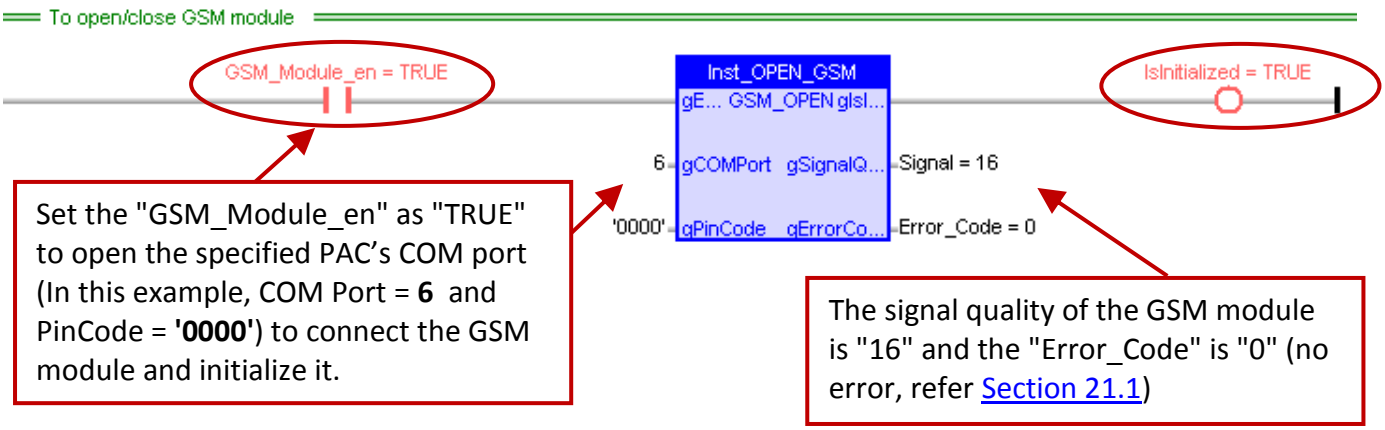
21.3 Test for SMS Messaging

In this example, we use one Win-GRAF XPAC and plug one 3G/2G module (I-8212W-3GWA + SIM card) in its slot1. Before testing this project, set the current IP address of the PAC in the Communication Settings, and then recompile and download this project to the Win-GRAF PAC. (Refer the Section 2.3.5 in case of any doubt.)

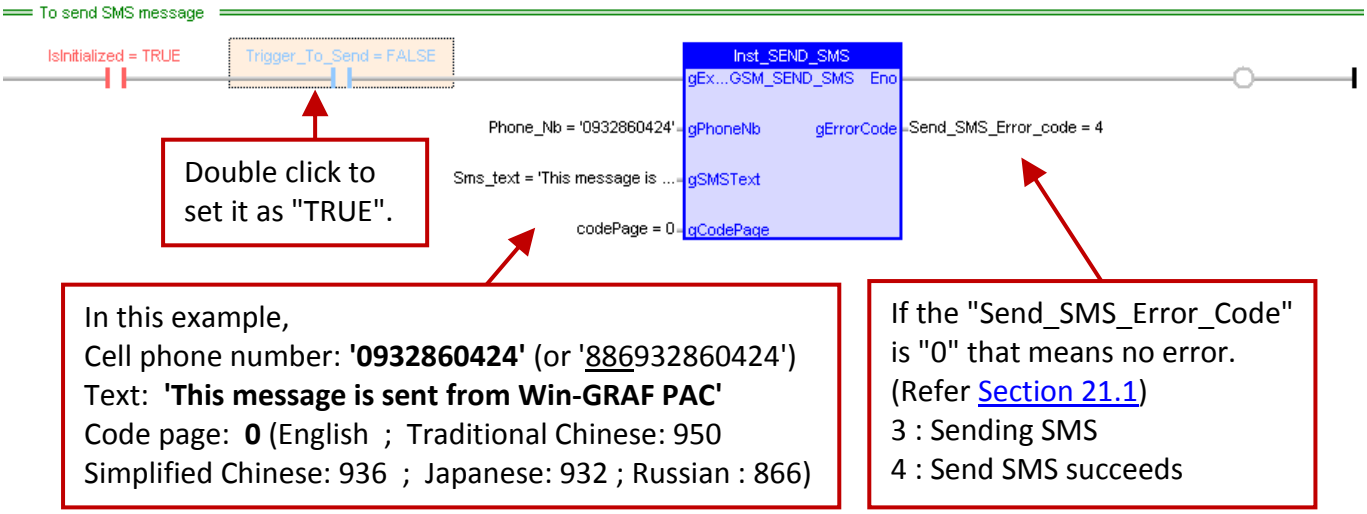
After connecting to the PAC, the "SMS_pro" program is shown as below:



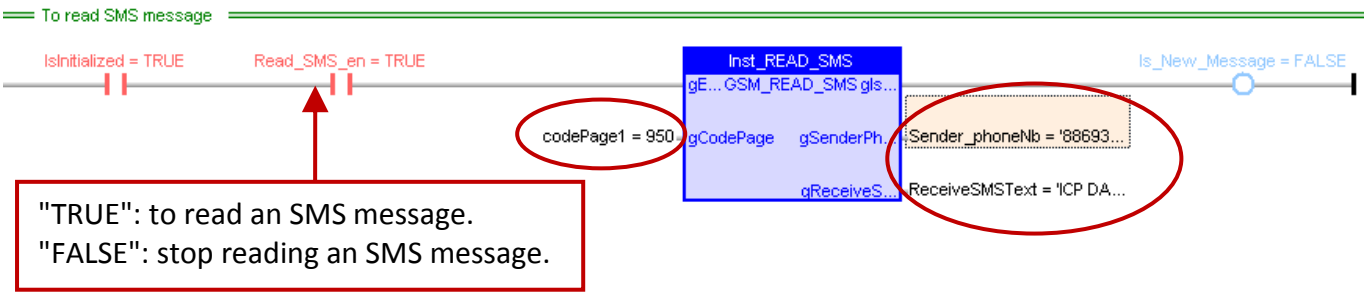
1. Fill in the COM Port No. which the GSM module used and the PIN code of your SIM card (if it is necessary to use). In this example, the GSM module uses the **COM6** (refer Section 19.1) and the SIM card PIN Code is **'0000'**.



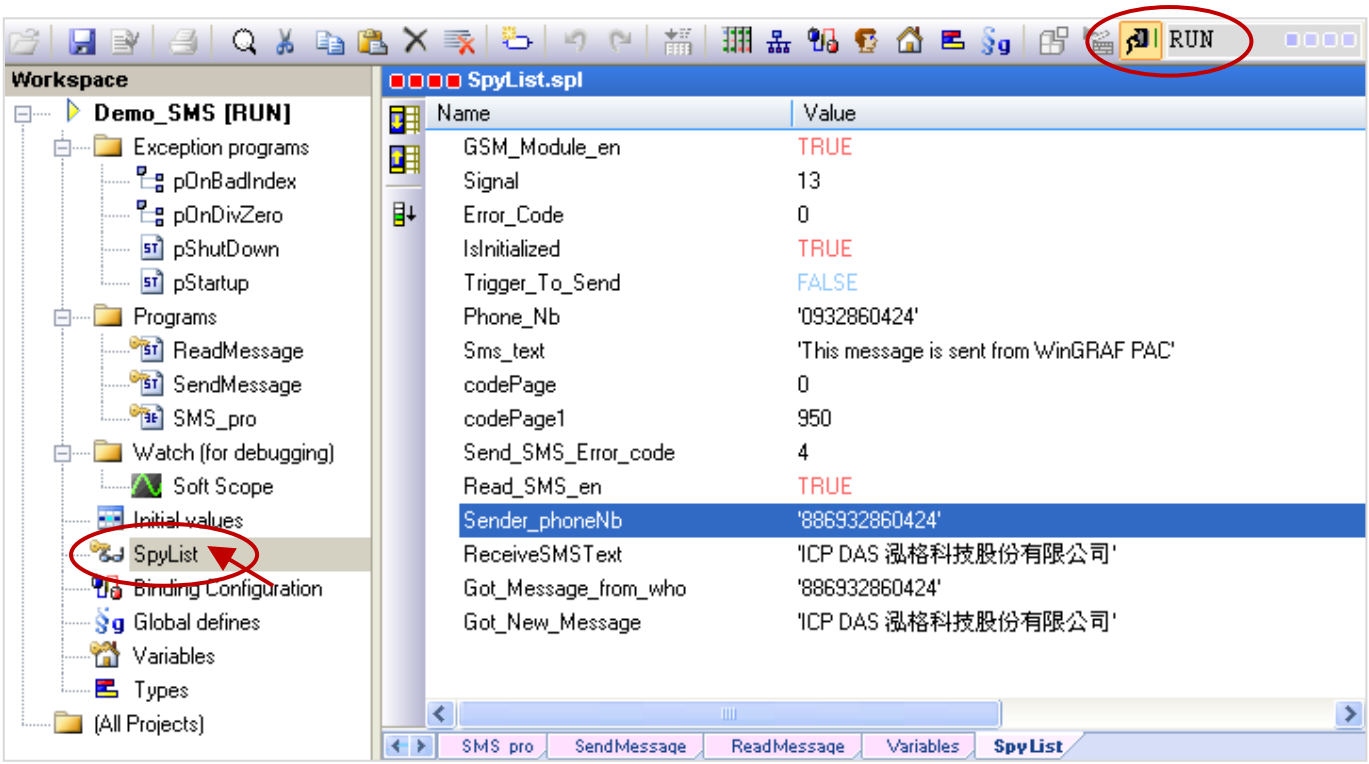
2. Fill in the phone number, the text message and the code page.
   In this example, the PAC will send an English (CodePage = **0**) text message (**'This message is sent from Win-GRAF PAC'**) to the cell phone number **'0932860424'** (or '886932860424').
3. Mouse double click the "Trigger_To_Send" to set it as "TRUE" to send a text message. If the "Send_SMS_Error_Code" changes to "4" that means the texting is successful (refer Section 21.1).



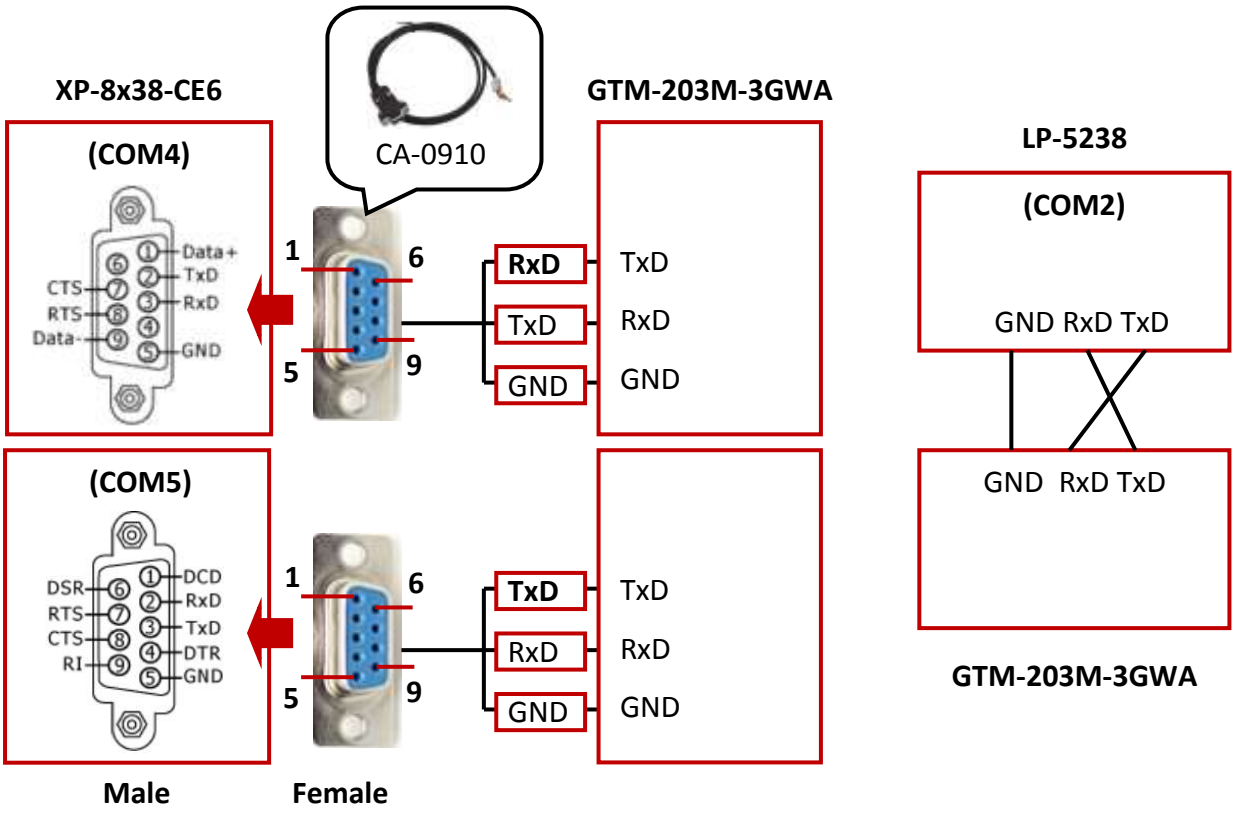
4. For testing an SMS message reading via the PAC, using a cell phone to send a text message to this SIM card. In this example, using the cell phone number (**'886932860424'**) to reply a Chinese (CodePage1 = **950**) text (**'ICP DAS 泓格科技股份有限公司'**) to this SIM card.



Moreover, the user can double click the "SpyList" to open the Spy list, and then change the variable value to test the SMS messaging.



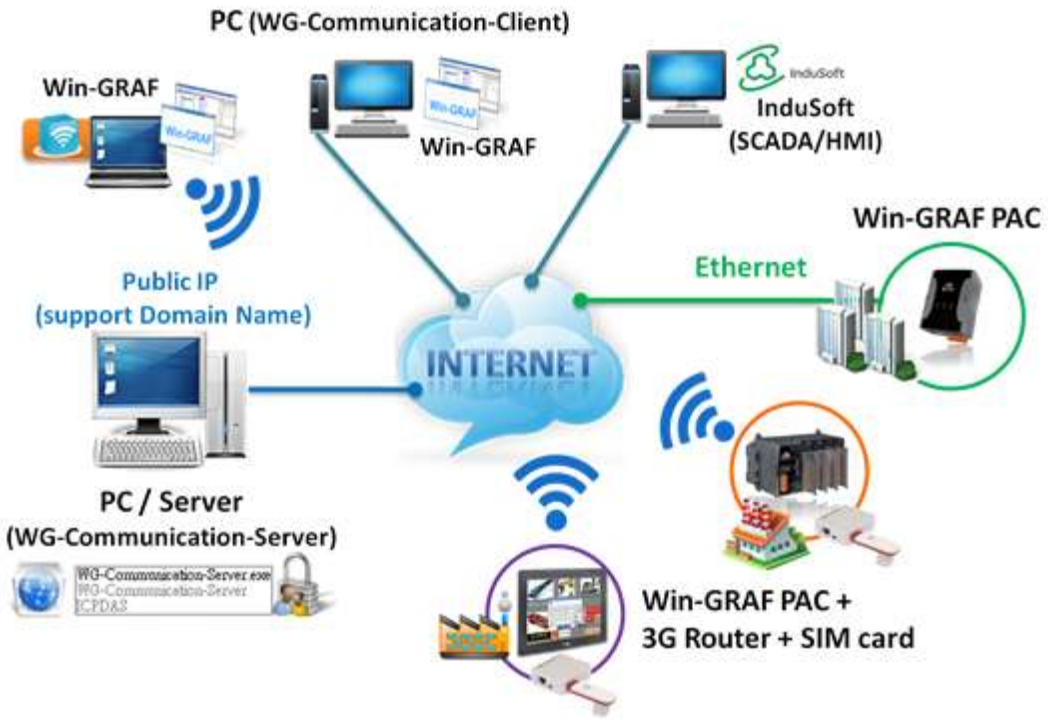
Note: If the 3G/2G wireless module is connected to the RS-232 Port of the Win-GRAF PAC, the user must refer the Appendix F to view the pin assignment of the COM Port. (Using the XP-8xx8-CE6 as an example, the pin assignment of the COM4 or COM5 is different.)



Chapter 22 The Intelligent Win-GRAF 3G Solution

The following Win-GRAF driver versions support intelligent 3G solution described in this chapter.

WP-8xx8 : 1.05 ; VP-x2x8-CE7 : 1.01 ; XP-8xx8-CE6 : 1.03 ; WP-5238-CE7 : 1.02



The main features of the intelligent 3G solution is to connect the Win-GRAF PAC to the Server (WG-Communication-Server) with a public IP (support Domain Name) via 3G wireless or via the Internet. Then the PC/laptop (WG-Communication-Client) can communicate with the remote PAC via the Server.

The user can achieve the following functions:

1. The user's PC/laptop can run the Win-GRAF Workbench to remotely debug/update the Win-GRAF application on the PAC.
2. The user's PC/laptop can run the SCADA/HMI software (e.g., InduSoft) to remotely monitor the PAC.
3. The user's PC/laptop can remotely update the Win-GRAF driver on the PAC if it is necessary.
4. The remote PAC can actively send a log file to the WG-Communication-Server.

Note:

1. The WG-Communication-Server must have a public IP address. Other remote PAC and user's PC/laptop no need a public IP address.
2. If Internet connection for the PAC is available, the user no need to buy a 3G Router.
3. If Internet connection for the PAC is unavailable, the user can buy a 3G Router and then insert a local SIM card to enable the PAC to access the Internet. We recommend you to choose an unlimited 3G data plan to reduce the Internet connection charges.
4. If there are several PACs in the same worksite to connect to the WG-Communication-Server, set up an Ethernet switch first. Then connect this switch to the 3G Router so that these PACs can access the Internet through this switch.

22.1 Set Up the PC/WG-Communication-Server

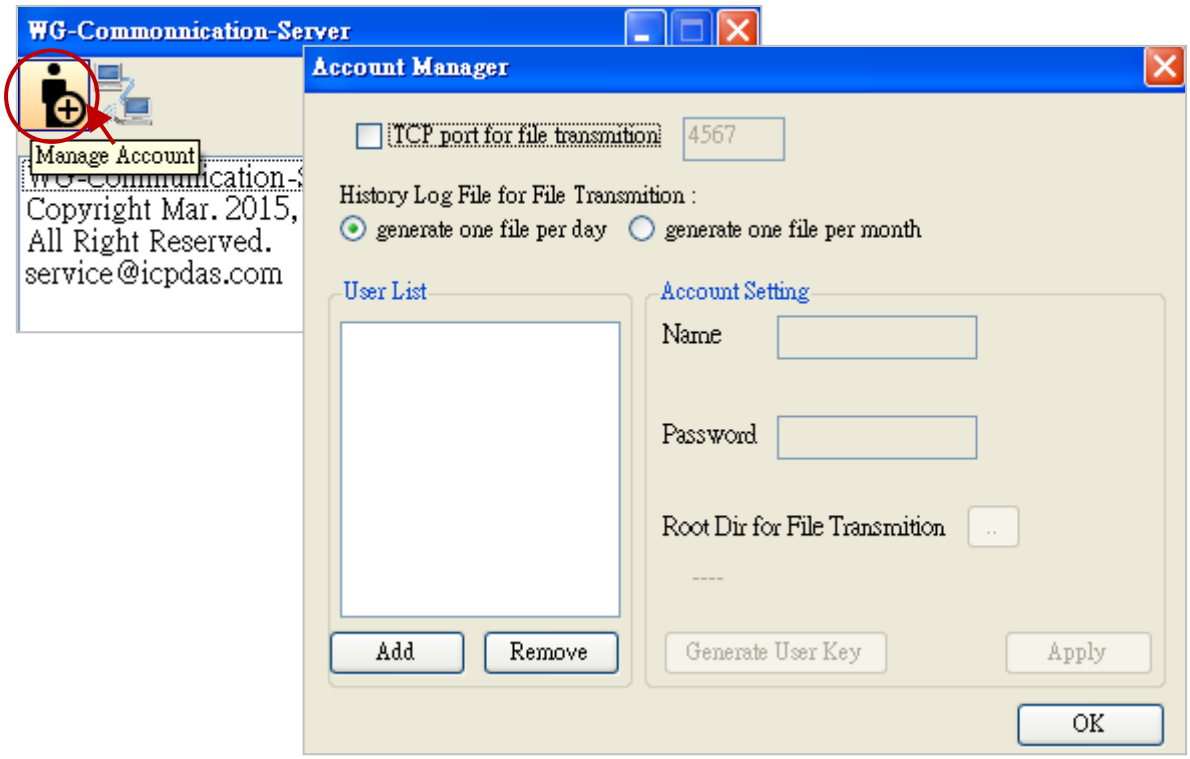
The user can use the "WG-Communication-Server" to create the user account (max. 100) for the remote PC/laptop or PAC to log in to the Server (**Note:** Both the PC/laptop and the PAC must log in with the same username/password, and the PC/laptop need to install the "WG-Communication-Client".) In the Win-GRAF PAC CD (CD-ROM: \Napdos\Win-GRAF\Tools_Utility), copy the "WG-Communication-Server" folder to your Server PC's "**D:**" (i.e., **D:\WG-Communication-Server**).

Notice: The "WG-Communication-Server.exe" must be stored in this folder to work properly.

Add a User Account:

1. After running the "WG-Communication-Server", it will zoom out to the bottom-right corner of your desktop screen (running in the background). Double-click the small icon if you want to configure it.

2. Click the "Manage Account" icon to open the setting window.

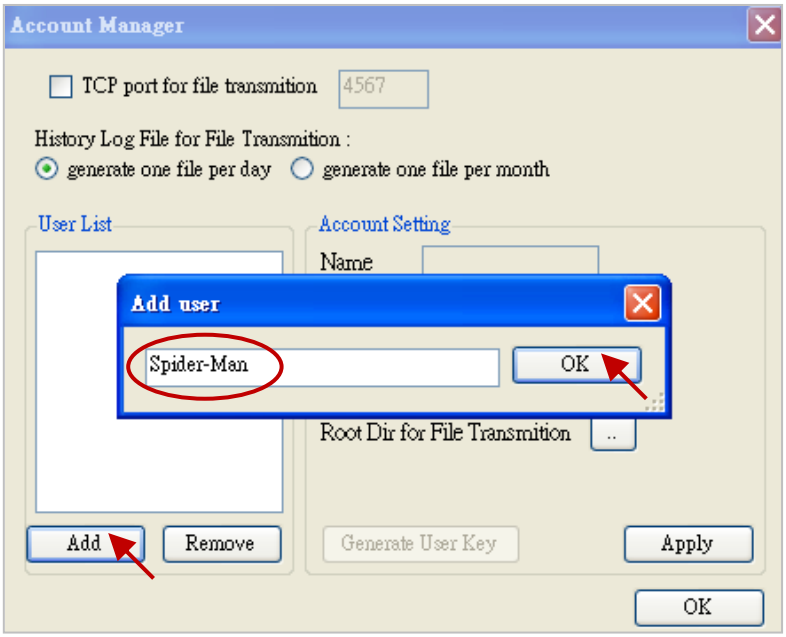


"TCP port for file transmission": To enable the specific TCP port (Default: 4567; Range: 1000 ~ 9999) for communicating with the PAC while file sending. (**Note:** Whether check or uncheck this item, or even modify the TCP Port number, the user must restart this software to apply the setting.)

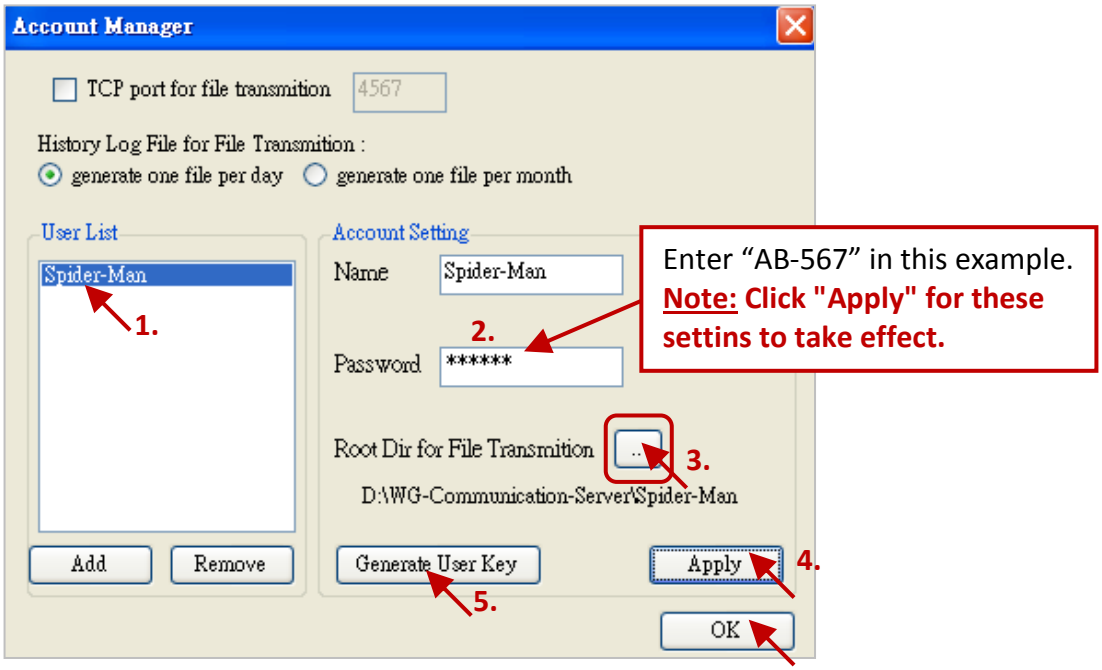
"History Log File": To generate one historical log file per day/month (defaults: "per day").

Note: Just go to the next step to set up an account. In addition, the user can open the "D:\WG-Communication-Server\account.txt" to check the account that you set up before.

3. Click the "Add" button to enter a username (e.g., "Spider-Man") and then click "OK".



4. Click on the username (e.g., "Spider-Man") to set its password (e.g., "AB-567"). Click "Root Dir" can set the storage path of the PAC file or user key, we recommend you to use the defaults - D:\ WG-Communication-Server\User Name (e.g., Spider-Man), and then click "Apply" to take effect (and unlock the "Generate User Key" button.



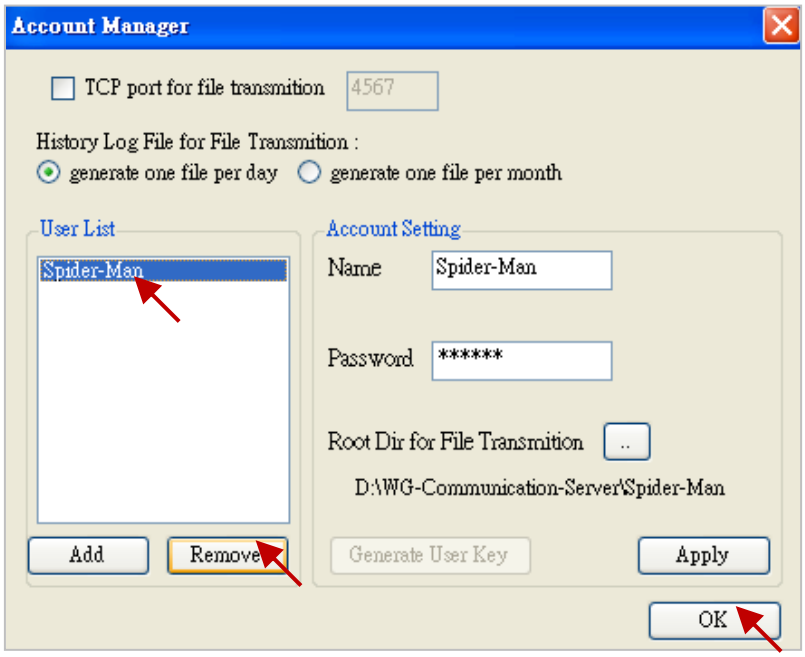
5. For security concerns, the remote PC/laptop must have a user key provided by the Server for successfully logging in to it. Click "Generate User Key" to create a user key.
Notice: Each time you click "Generate User Key" and confirm it, a new user key will be created. **Be careful to make sure you want to do it because the PC/laptop which includes an old user key cannot login to the Server any more** (unless installing the generated new user-key in it) .

If it must to create a new user key, click "Yes", and then the old user key on remote PC will become invaild.

Send the user key to the remote user (client), and copy it to the **D:\WG-Communication-Client** on user's PC. Then, the user will have the access authority to use this username/password to login to the Server.

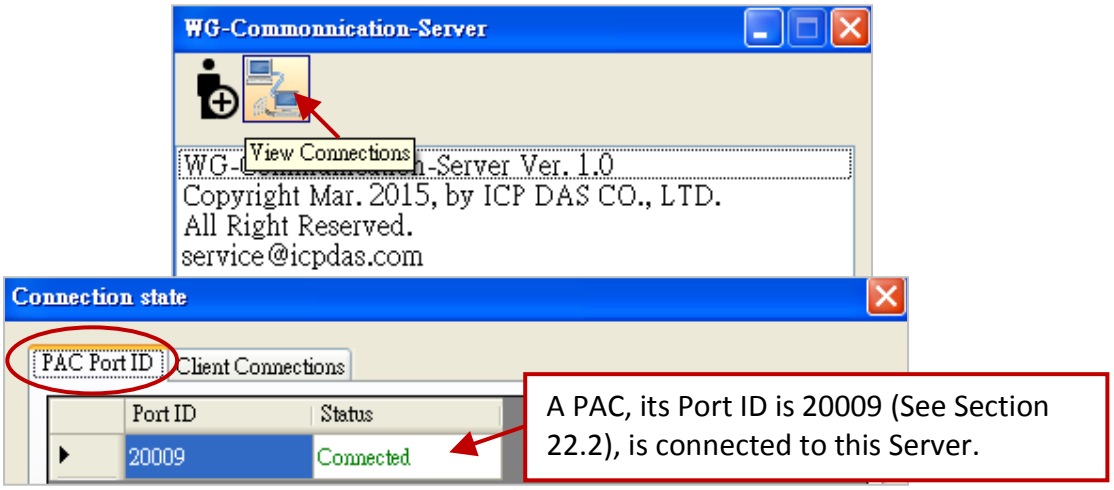
Delete a User Account

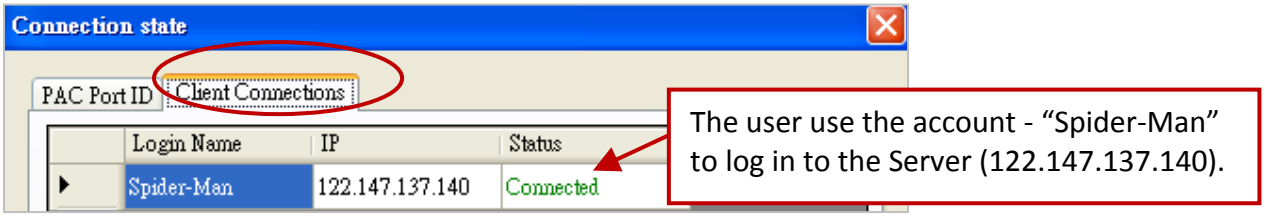
Click on the username you want to delete (e.g., "Spider-Man"), click the "Remove" button, and click "OK" to delete this username/password.



View the connection status

This feature is used to view what PC/laptop or Win-GRAF PAC is connected to this Server.





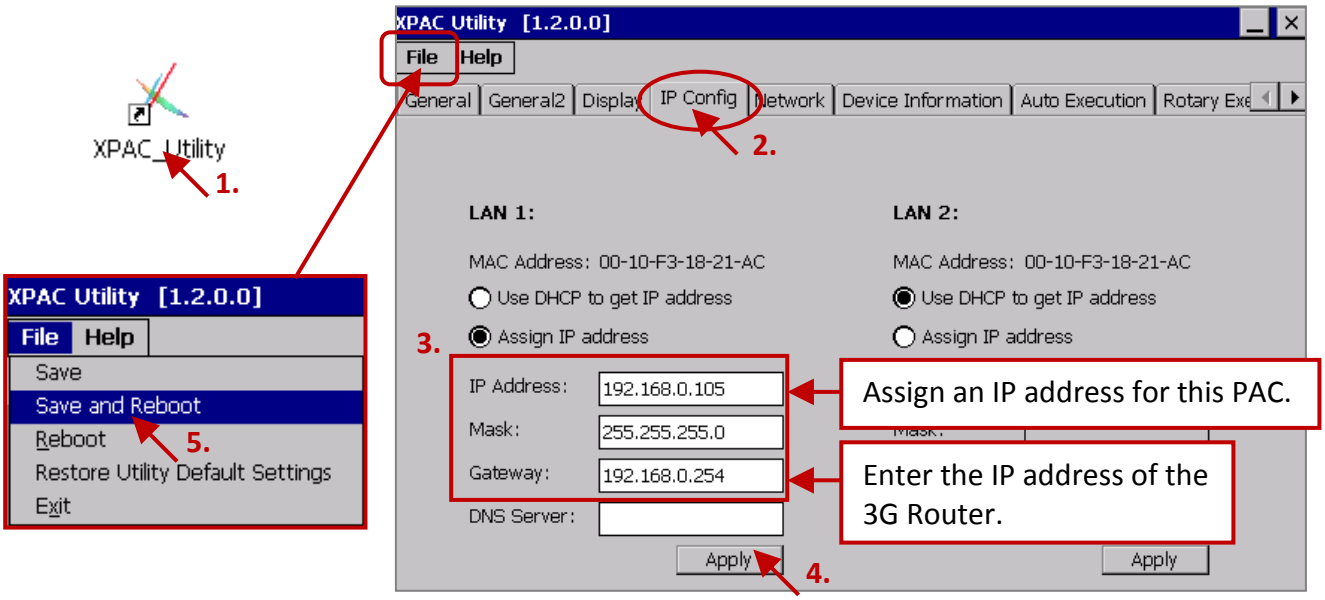
22.2 Set Up a Remote PAC to Connect to the WG-Communication-Server

If the Win-GRAF PAC connects to the Internet via a 3G Router (plus a SIM card), see its product instructions to configure it. **Note:** Many 3G routers on the market has the build-in Wi-Fi feature. For security issues on PAC communication data, we recommend you to disable the Wi-Fi function of the 3G Router.

The network setting on the PAC

Please configure the network settings according to the field demands. If the user uses the 3G Router as a gateway and set its IP address as "192.168.0.254", and then assign IP addresses which range from "192.168.0.100" to "192.168.0.200" for PACs, then you can set the PAC's network setting as below.

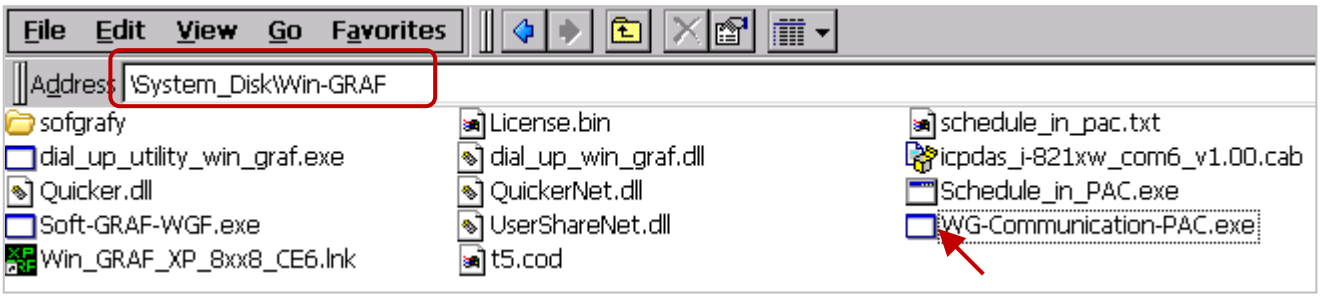
- Double click the PAC Utility (e.g., XPAC_Utility) on the PAC's desktop, and then click the IP setting tab (e.g., IP Config) to fill in the proper IP, Mask, and Gateway address, and click "Apply". Finally, execute "File → Save and Reboot" to save and reboot the PAC.



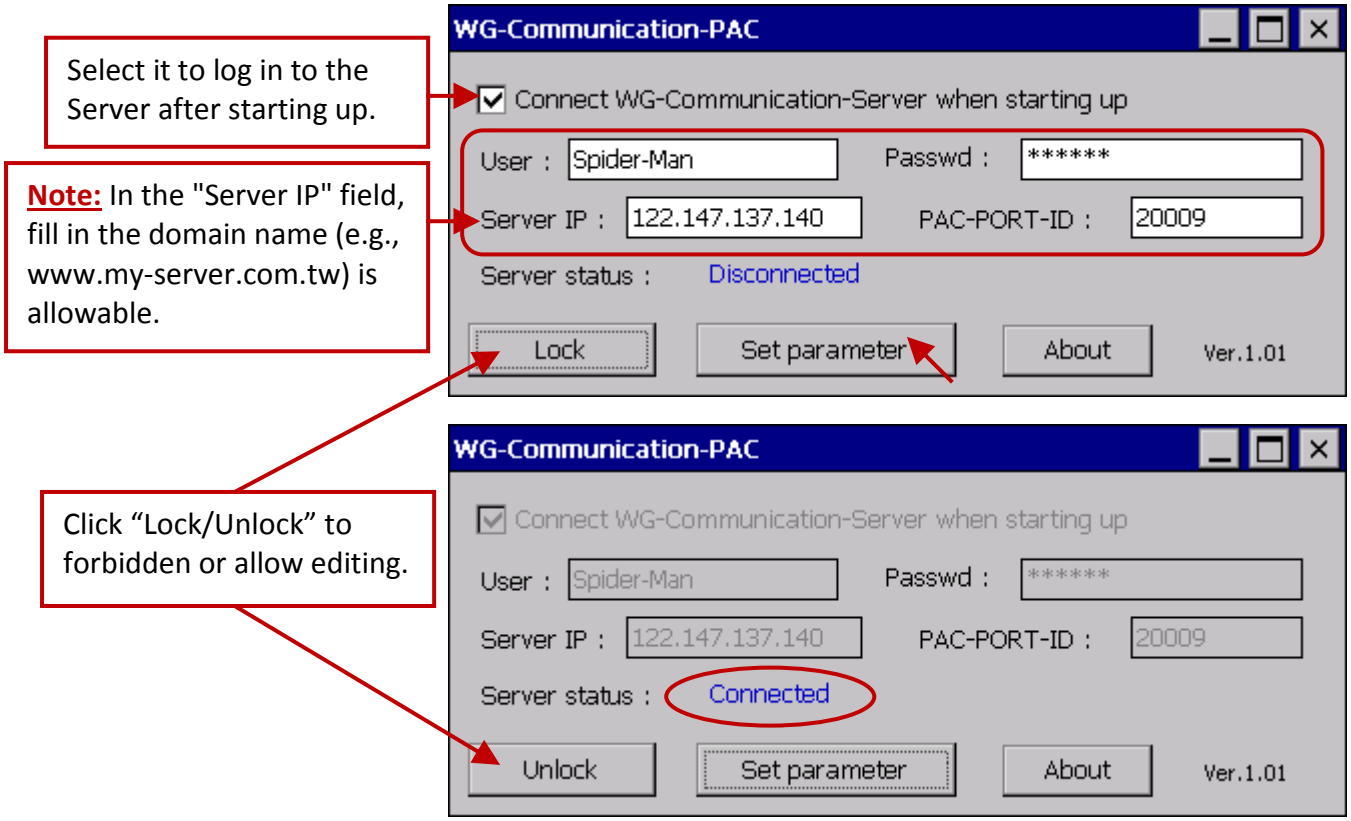
Using the "WG-Communication-PAC" on the PAC to connect to the Server

On the PAC, you can find out "WG-Communication-PAC.exe" at the path "\System_Disk\Win-GRAF\", and then use it to connect to the WG-Communication-Server.

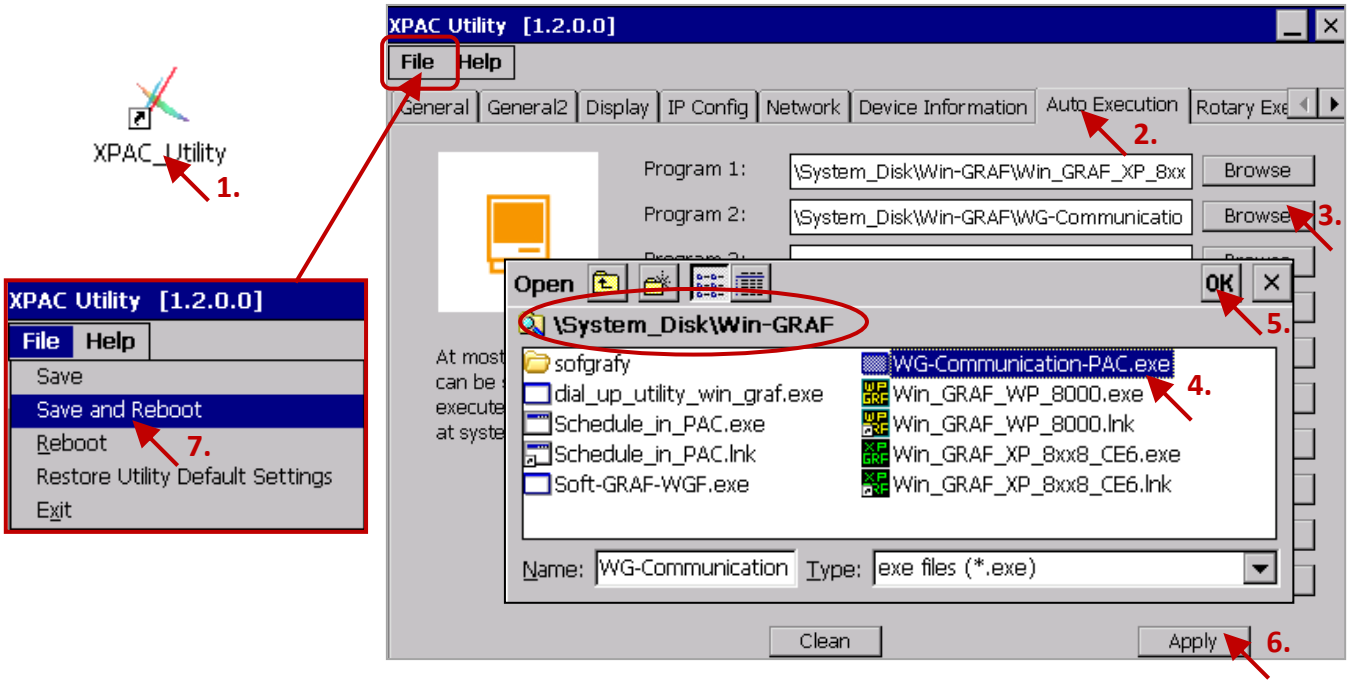
1. Double click the "WG-Communication-PAC.exe" to open the setting window.



2. Fill in the username/password (e.g., Spider-Man/ AB-567, refer Section 22.1) that created by the WG-Communication-Server, and enter the IP address of the Server, and then assigns a Port ID (e.g., 20009; Range: 20000 ~ 22000) for this PAC. (**Note:** The "PAC-Port-ID" is used for a remote Server or a user's PC to identify which PAC is connected with them. Different PACs connect to the same Server should use different PAC-PORT-ID.)
3. Finally, click the "Set parameter" button to take effect.



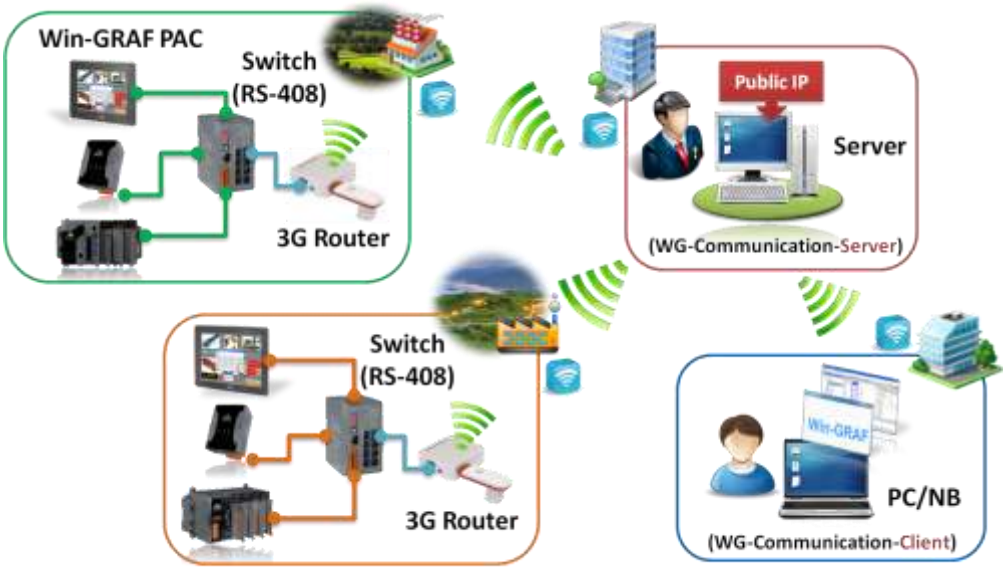
Moreover, run the PAC Utility (e.g., XPAC_Utility) and then follow the steps (shown as below) to add "WG-Communication-PAC.exe" into the startup program list, and then execute "File → Save and Reboot" to save and reboot the PAC.



22.3 Set Up the WG-Communication-Client on a User PC or the SCADA PC

The user's PC can connect to the Server and then debug/download the Win-GRAF project or update the Win-GRAF driver to the remote PAC by using the "WG-Communication-Client" software. In the Win-GRAF PAC CD (CD-ROM: \Napdos\Win-GRAF\Tools_Utility), copy the "WG-Communication-Client" folder to your PC's "**D:**" (i.e., **D:\WG-Communication-Client**).

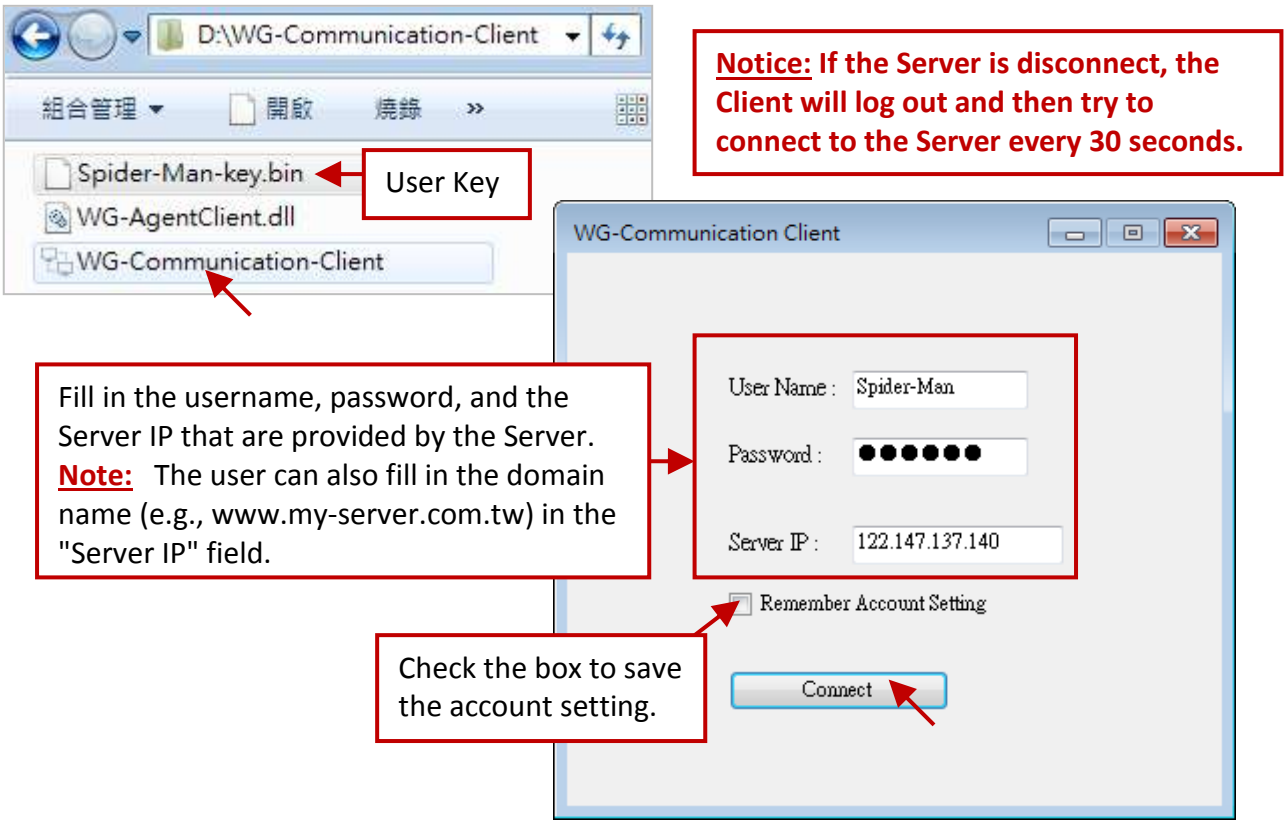
Notice: The "WG-Communication-Client.exe" must be stored in this folder to work properly.



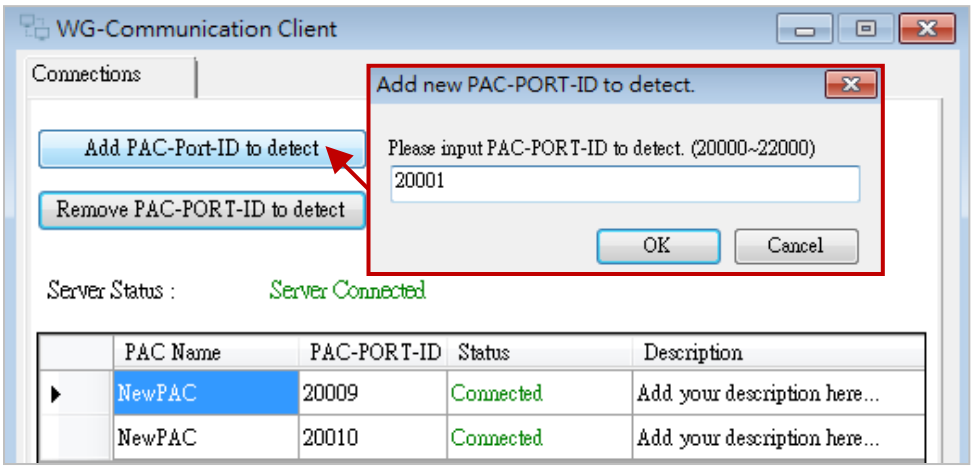
22.3.1 Logging to the Server and detect the PAC connection status

Before logging to the Server, make sure the "WG-Communication-Client" folder contains the user key provided by the Server.

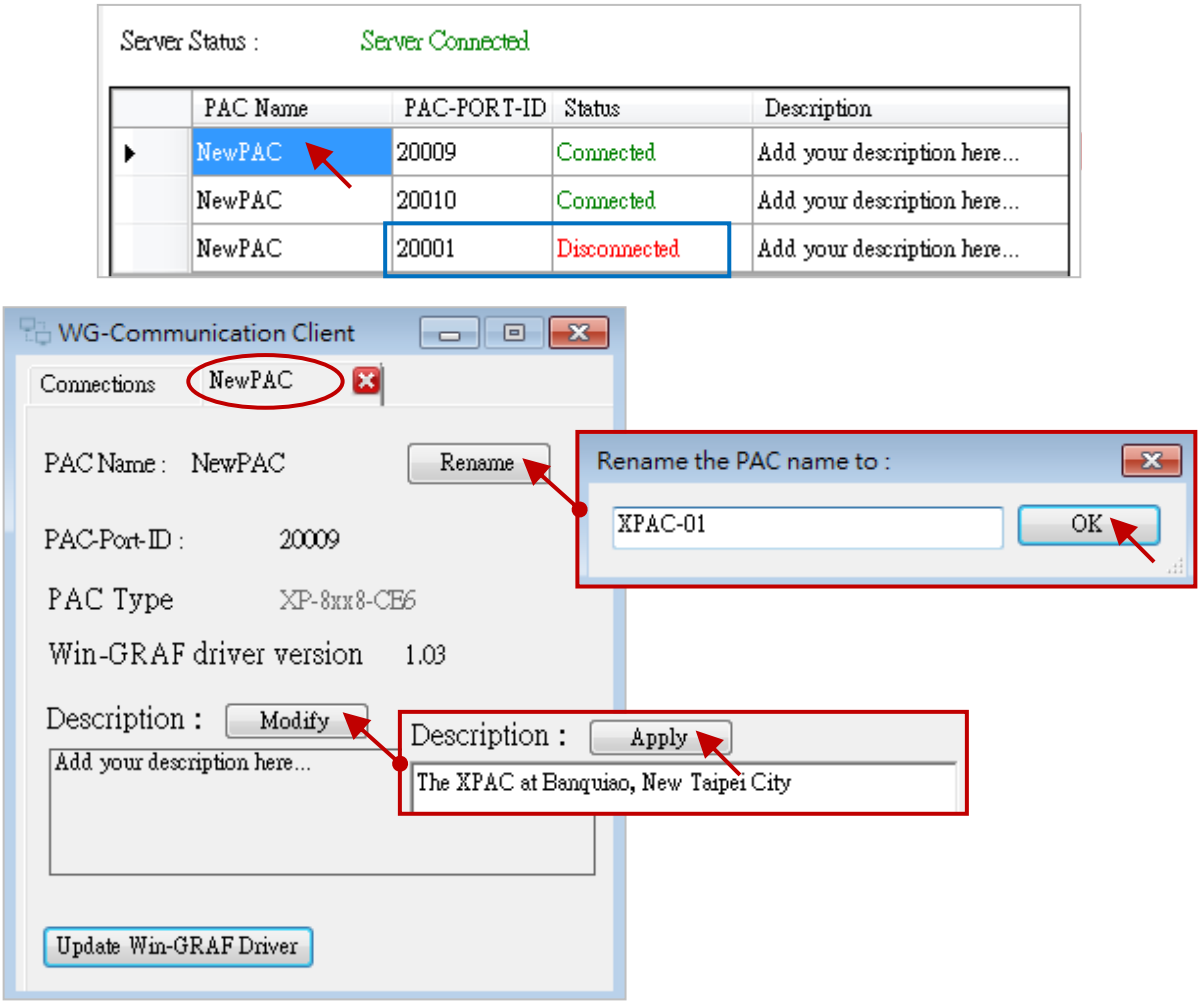
1. Mouse double click the "WG-Communication-Client" to open the setting window, and fill in the user name, password, and Server IP, and then click the "Connect" button.



2. After connecting to the Server, you can see the "Server Status" shows "Server Connected" and the following table will list the status of the PAC connection that connect with the Server (refer Section 22.2). The user can click "Add PAC-Port-ID to detect" to add a PAC-PORT-ID (e.g., 20001) to the PAC connection-detection lists. Or, click "Remove PAC-Port-ID to detect" to remove it from the lists.



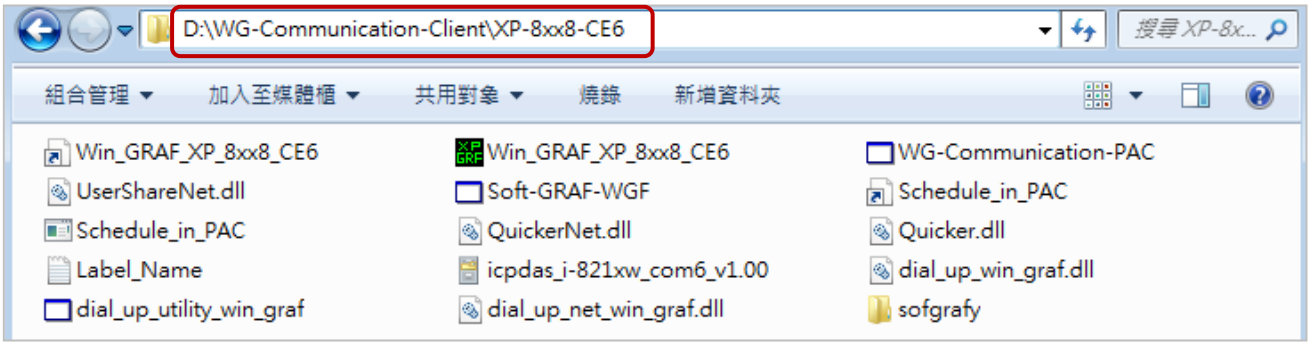
3. Mouse double click on any PAC name to open that setting tab. The user can click “Rename” to change the displayed PAC name, and also can click “Modify” to add comments, and then click “Apply” to take effect.



Remotely update the Win-GRAF PAC Driver from the PC

In the normal case, the user doesn't need to update the Win-GRAF PAC driver unless you want newer driver version that contains new released functions or bug fixing. The user can visit the website to download the latest version of the Win-GRAF PAC driver.
www.icpdas.com/root/product/solutions/softplc_based_on_pac/win-graf/download/win-graf-driver.html

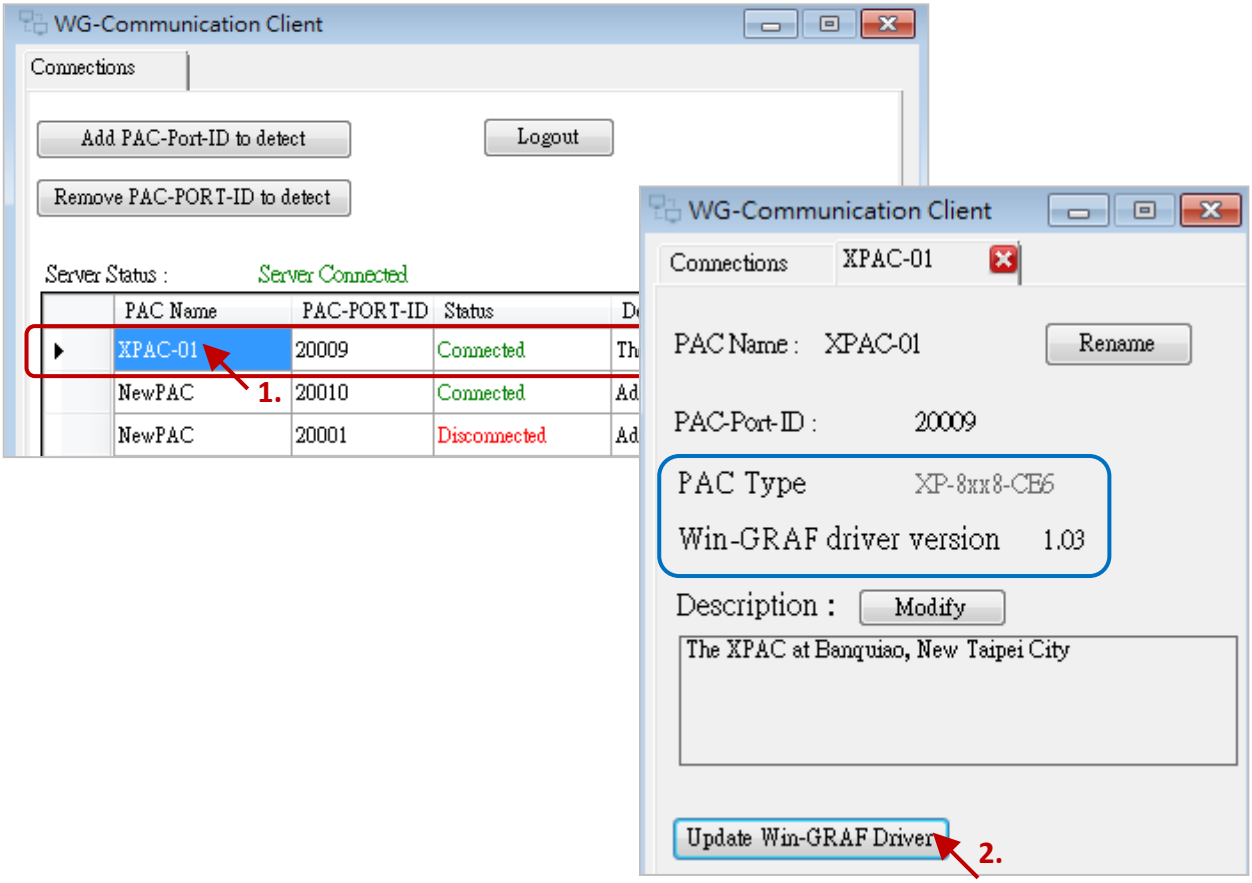
Before updating the Win-GRAF driver, make sure you have copied all related files into the proper folder. Using the "**XP-8xx8-CE6**" as an example, first unzip the driver file (.zip) and then copy all files from the \xp-8xx8-ce6-driver-1.xx\1.xx (e.g., 1.03) to the D:\WG-Communication-Client**XP-8xx8-CE6**.



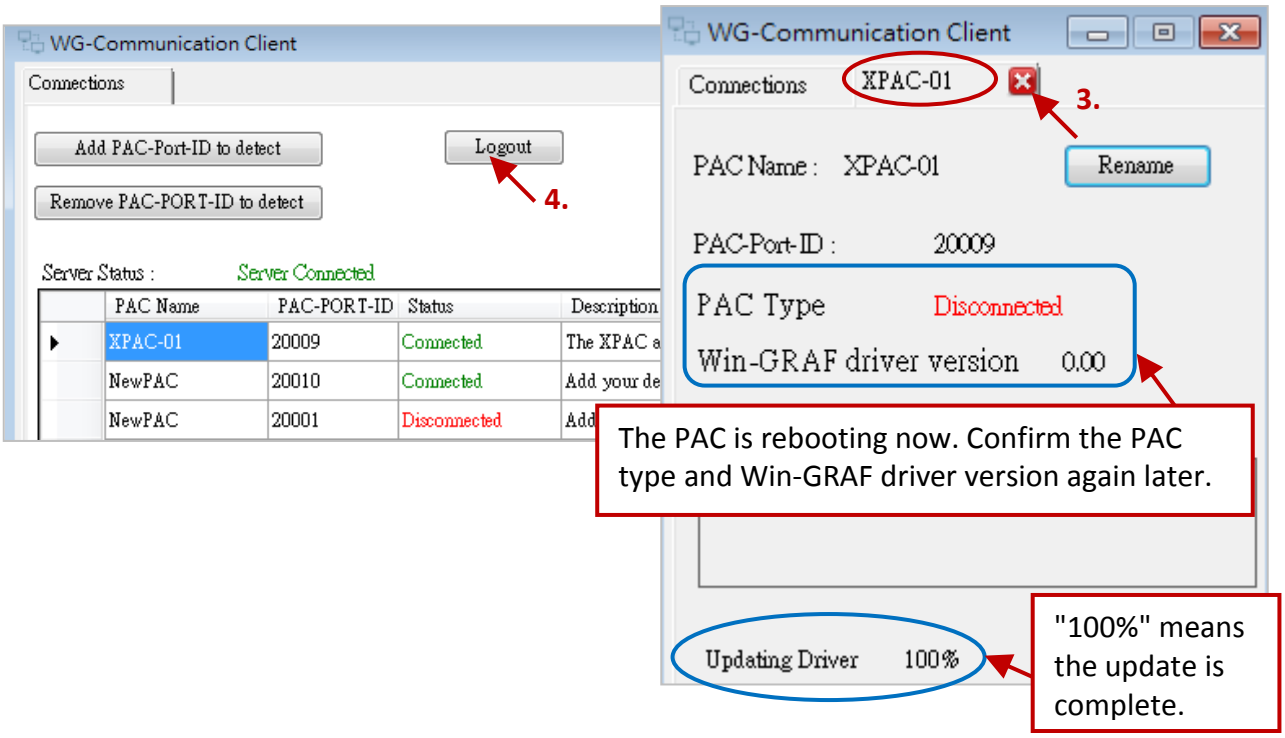
1. Run the D:\WG-Communication-Client\WG-Communication-Client.exe and log in to the Server. (If you are not familiar with the way, refer Section 22.3.1.)

2. Double click on the PAC name that you want to update and has been connected to the Server, and then click the "Update Win-GRAF Driver" button.

   Note: Don't restart this PAC, stop this Client, or close this tab while updating the driver. And, update only one PAC at each time.



3. When the “Updating Driver” shows “100%” means that the update is complete. Waiting for about 60 to 90 seconds. If the “PAC Type” shows “Disconnected” which means the PAC is rebooting now. After that, confirm the PAC type and Win-GRAF driver version again.
4. Close the “XPAC-01” tab. Then, you can change the other PAC’s settings or click “Logout” to log out the Server.

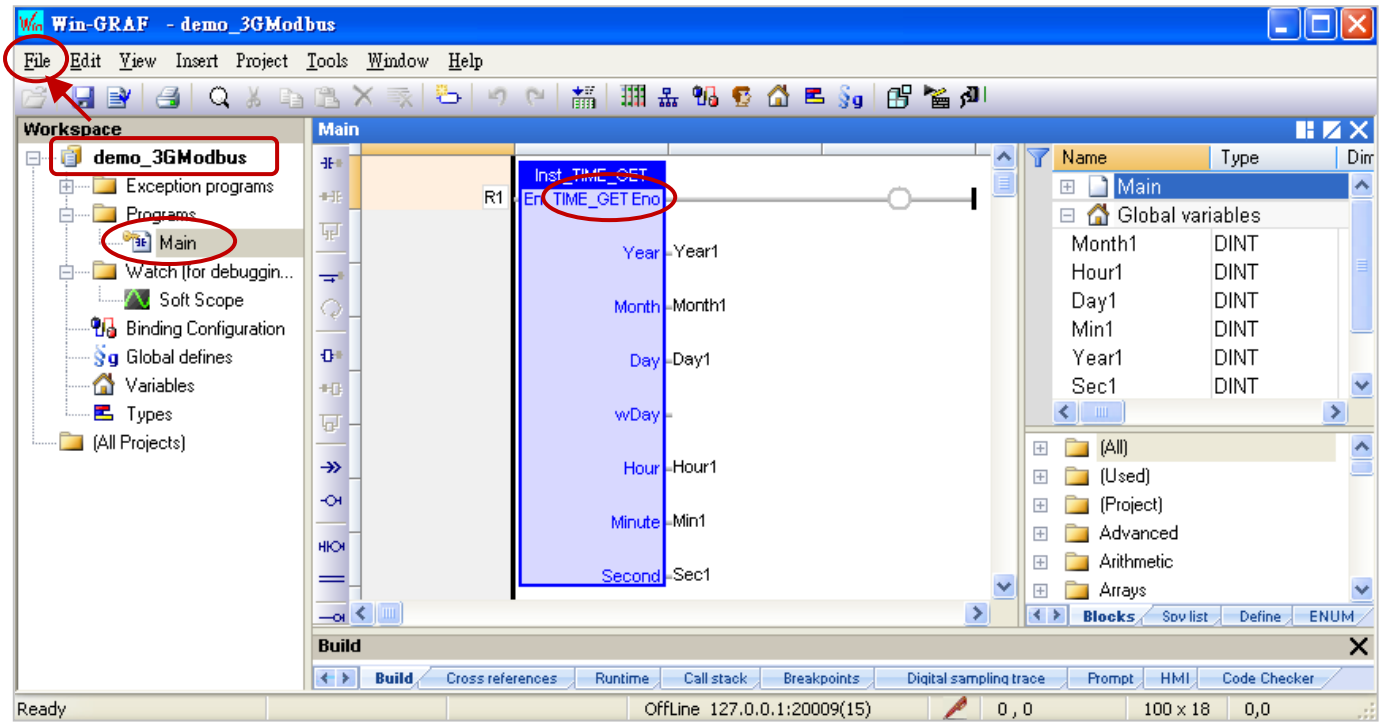


22.3.2 Remotely update the Win-GRAF project from the PC/Win-GRAF Workbench

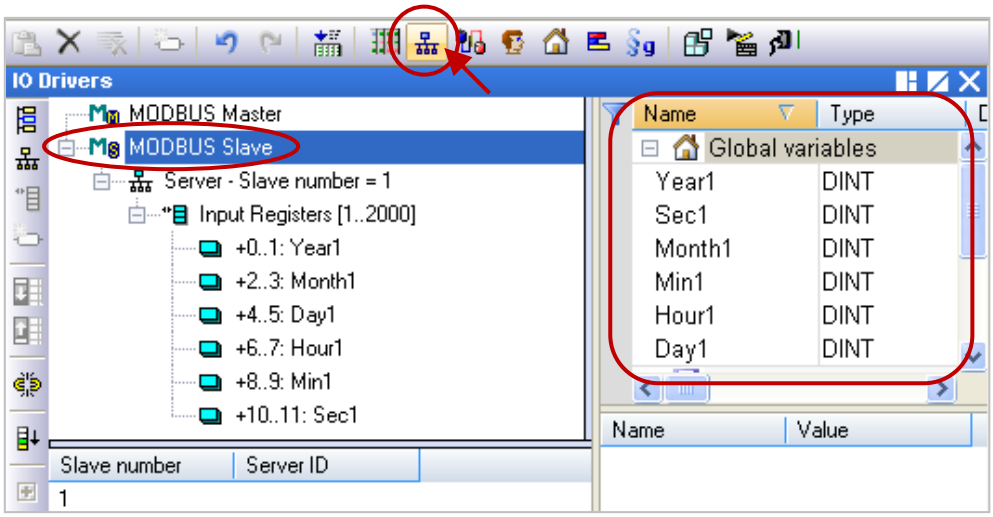
This section will describe how to use the WG-Communication-Client on the PC/Win-GRAF Workbench to connect to the Server and then remotely download/update the Win-GRAF project to the PAC.

1. Run the D:\WG-Communication-Client\WG-Communication-Client.exe and log in to the Server. (If you are not familiar with the way, refer Section 22.3.1.)

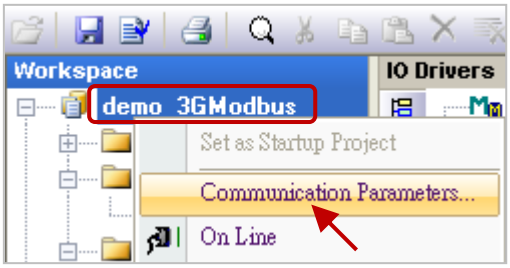
1. Run the Win-GRAF Workbench and then click "File → Add Existing Project → From Zip..." to open the Win-GRAF project (demo_3GModbus.zip) which is stored in the Win-GRAF PAC CD and its file path is CD-ROM: \Napdos\Win-GRAF\demo-project\. This example project includes an LD program (Main), using the "Time_Get" function block to read the system time.



And, this project enables a Modbus TCP Slave to allow the SCADA/HMI software (e.g., "InduSoft") or the Modbus Master to access Win-GRAF variables (refer Section 3.1 for more details).



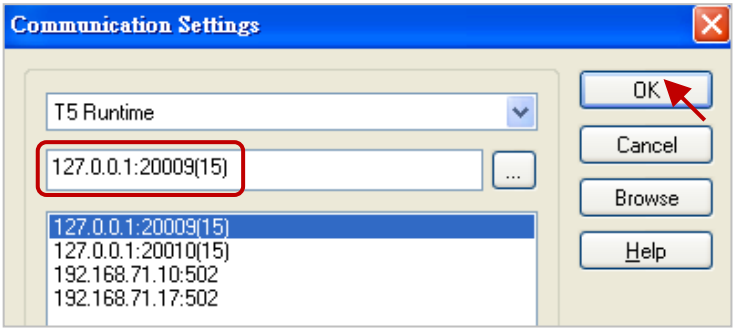
2. Mouse right click on the project name (demo_3GModbus), and then click "Communication Parameters.." to enter the IP address and the port number.



The configure way is "**127.0.0.1:PAC-PORT-ID(Timeout)**", and it's set to "127.0.0.1:20009(15)" in this example.

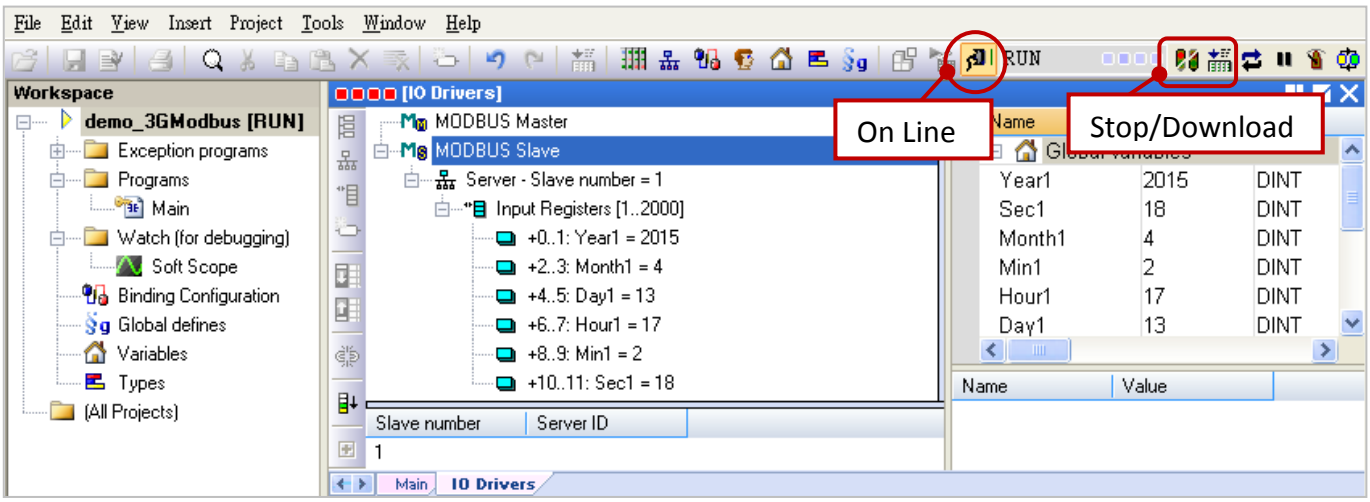
Note: "PAC-PORT-ID" is a uniquely identified ID, and make sure this PAC is connected to the Server (refer Section 22.2). By default, the Timeout is 3 seconds, recommend you to set it between 15 and 30 seconds if the PAC access to the Internet via the 3G router (plus a SIM card).

Due to the 3G is a wireless network and transmit signals through a base station, it can cause communication delays or exceptions sometimes. Recommend you to set a bigger timeout value to range from 15 to 30 seconds to reduce the communication timeout or read/write error.



3. Click the "On Line" button to make a connection and download the project to the PAC.

Note: If you ever edit this program, it must be compiled before downloading (refer Section 2.3.4); If there is the other project name is running, just click "Stop" to stop that project and then click "Download" to download the current project (refer Section 2.3.5 - Steps 4 ~ 6).



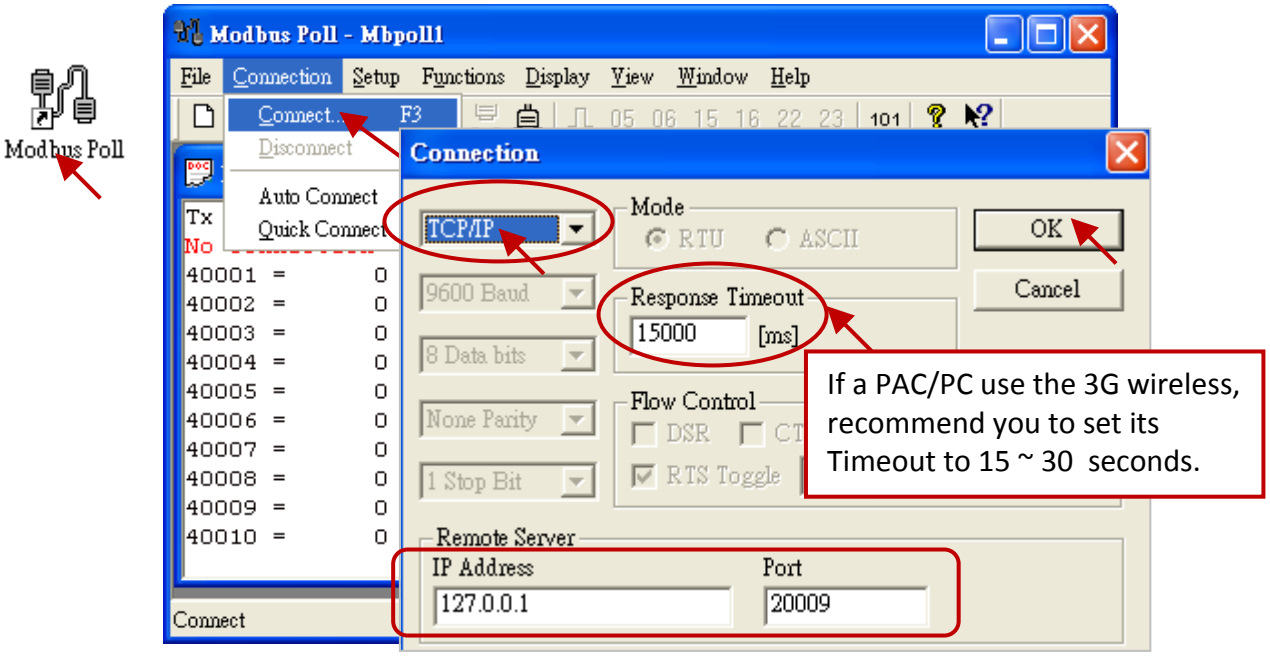
22.3.3 To read/write the PAC data remotely from the PC/SCADA

For the different kinds of HMI/SCADA software intend to communicate with a remote PAC via Modbus TCP protocol through the WG-Communication-Server, the user must run the WG-Communication-Client on an HMI/SCADA PC first. Then use it to log in to the Server, and then enter the PAC's IP address (fixed 127.0.0.1) and PAC-PORT-ID (20000 ~ 22000) that you want to connect to on the SCADA software.

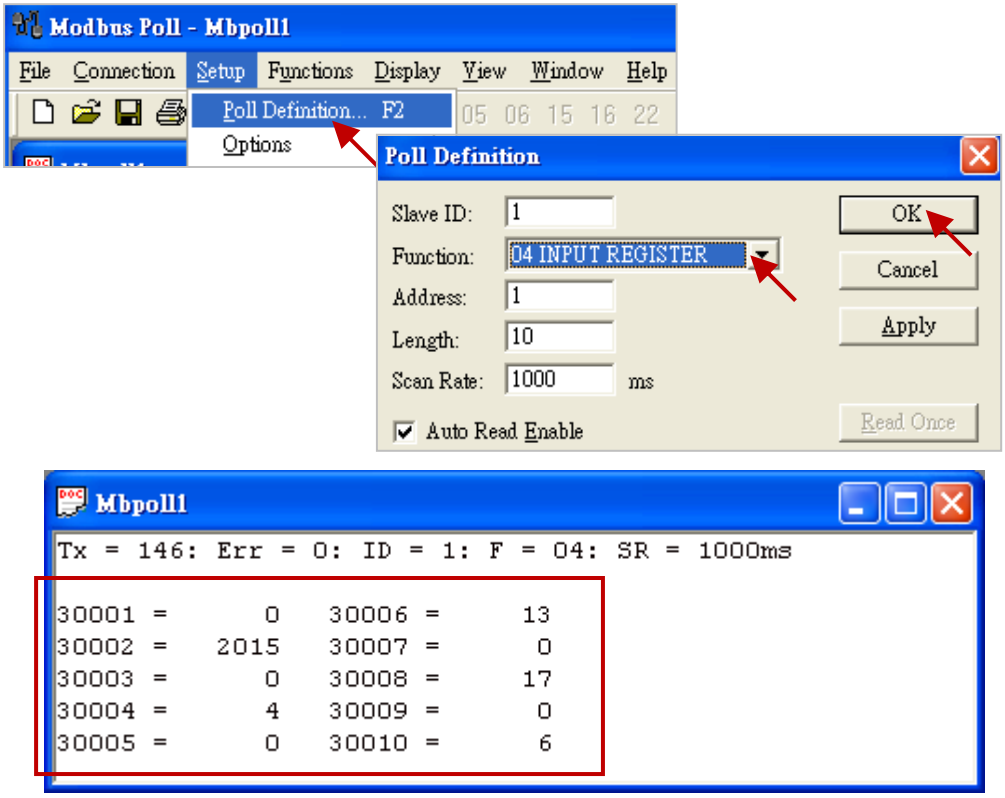
In this section, we will introduce you how to use the Modbus Poll to act as an HMI/ SCADA/Modbus Master on the PC to read the PAC data remotely. You can visit the website to download and install the Modbus Poll: http://www.modbustools.com/modbus_poll.html

1. First, run the “D:\WG-Communication-Client\WG-Communication-Client.exe", and then log in to the Server. (If you are not familiar with the setting, refer Section 22.3.1)

2. Double click the “Modbus Poll” icon to open it, and then click “Connection → Connect” to set the connection parameters. Select the “TCP/IP” way to connect, set the "Response Timeout” ranges from 15 to 30 seconds, set the "IP Address" as “127.0.0.1” (fixed), and set the "Port" as the PAC’s ID (PAC-PORT-ID, e.g., 20009, refer Section 22.2).



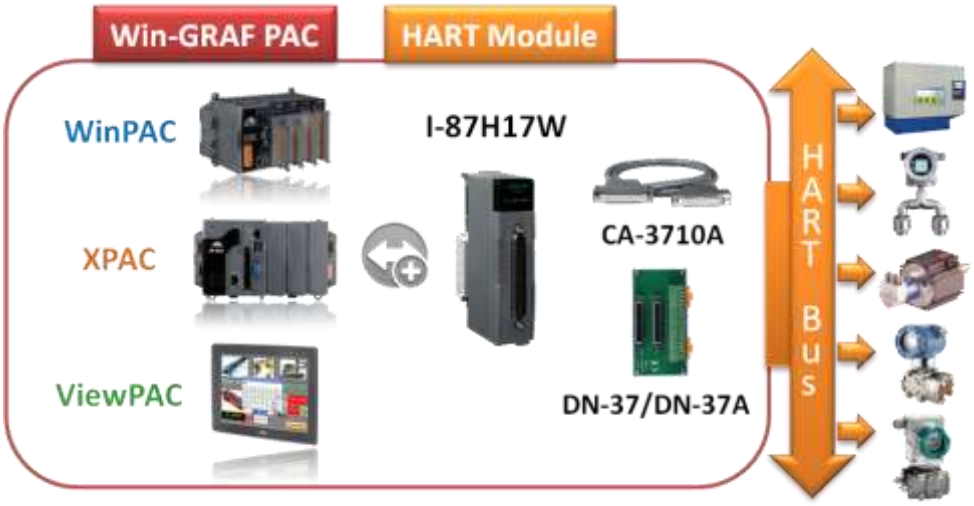
3. Click “Setup → Poll Definition” to set the data type you want to read. Then, select “04 INPUT REGISTER” in the "Function" field (refer Section 22.3.2 – Step2) and click “OK” to view the read data shown on the Modbus Poll.



Chapter 23 HART Master

The following Win-GRAF driver versions support the I-87H17W HART Master module.

WP-8xx8 : 1.07 ; XP-8xx8-CE6 : 1.05 ;



This chapter will describe three demo projects for communicating between the Win-GRAF PAC and the HART device via the I-87H17W HART module.

23.1 Introduction of the I-87H17W HART Module

I-87H17W : http://www.icpdas.com/products/Remote_IO/can_bus/i-87h17w.htm

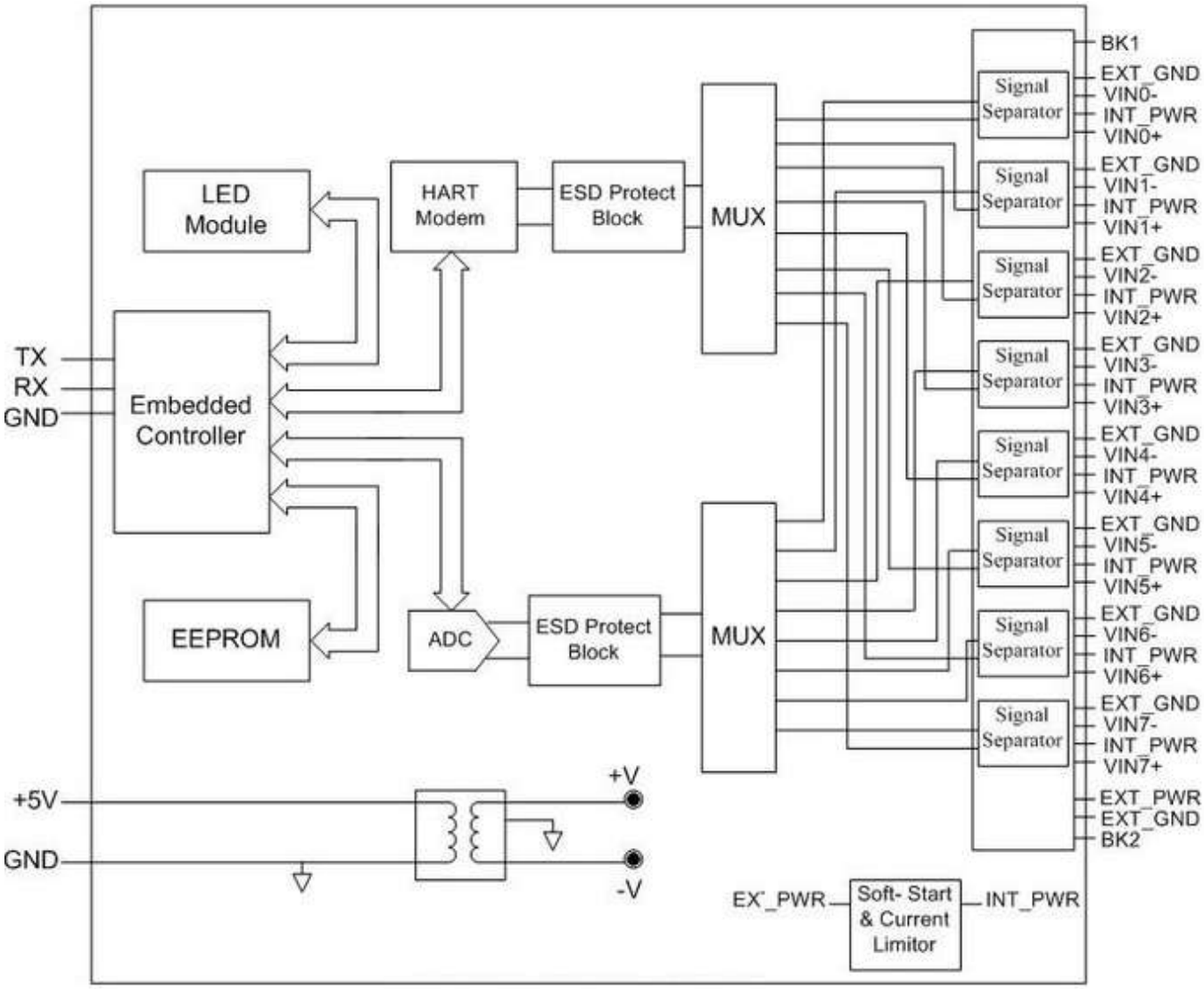
The XP-8xx8-CE6 PAC supports I-87H17W in its slot 1 to 7 (the first I/O slot No. is 1). And, the WP-8xx8 PAC supports I-87H17W in its slot 0 to 7. The Win-GRAF PAC doesn't support the I-87H17W which is plugged into the RS-485 remote expansion unit (like the I-87K8, RU-87P8), so only use it in the Win-GRAF PAC's I/O Slot.

Each I-87H17W has 8 analog input channels that used to measure 4 ~ 20 mA current and/or used as HART communication channels. Recommend to link only one HART device in each channel. Owing to these eight HART channels share with one communication chip inside this module, only single channel can be used at a time. In other words, I-87H17W unable to send/receive the HART frame via two or more channels at the same time. But, the user can use the Win-GRAF program (e.g., "Demo_HART_3.zip") to process these 8 HART channels to communicate with HART devices by turns.

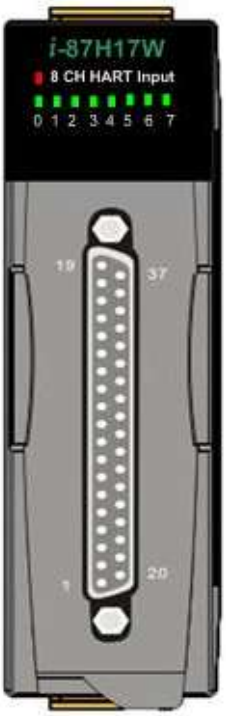
Hardware Wiring:

The user can visit the web page for more details on I-87H17W hardware.
http://www.icpdas.com/products/Remote_IO/can_bus/i-87h17w.htm
The following only lists its internal I/O structure, pin assignment and wire connection.

The internal I/O structure of I-87H17W:



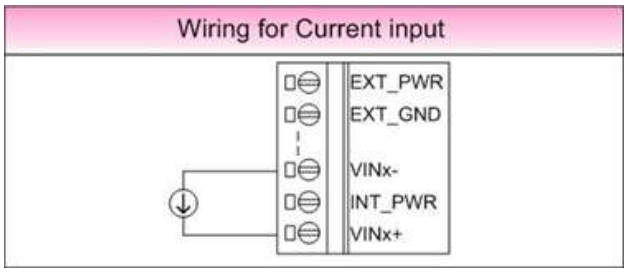
The pin assignment of I-87H17W:

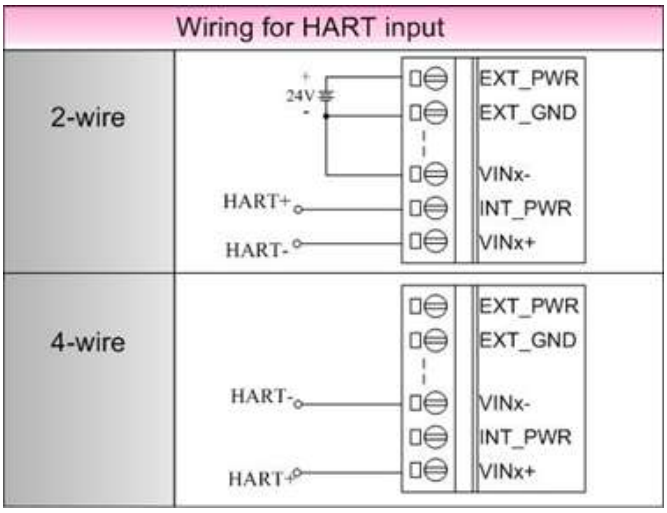


Pin Assignment Name	Terminal No.		Pin Assignment Name
X	19	37	BK2
EXT_PWR	18	36	EXT_GND
VIN7-	17	35	INT_PWR7
VIN7+	16	34	EXT_GND
VIN6-	15	33	INT_PWR6
VIN6+	14	32	EXT_GND
VIN5-	13	31	INT_PWR5
VIN5+	12	30	EXT_GND
VIN4-	11	29	INT_PWR4
VIN4+	10	28	EXT_GND
VIN3-	09	27	INT_PWR3
VIN3+	08	26	EXT_GND
VIN2-	07	25	INT_PWR2
VIN2+	06	24	EXT_GND
VIN1-	05	23	INT_PWR1
VIN1+	04	22	EXT_GND
VIN0-	03	21	INT_PWR0
VIN0+	02	20	EXT_GND
BK1	01		

37-pin male D-Sub Connector

The wire connection of I-87H17W:





23.2 The Format of the HART Protocol

Before describing the Win-GRAF projects, we need to understand the format of the HART protocol. The HART protocol is based on a Master-Slave communication scheme, using a "request-respond" way to communicate to each other for data exchanging.

Notice:

A. Refer the user manual of each HART device for knowing their supported HART format.
B. The user can ignore the “Check Byte” in the Win-GRAF program because the I-87H17W module will automatically add/remove it when sending/receiving a HART frame.
C. The HART physical layer is using 1200bps, 1 start-bit, Odd parity, 8 character-size and 1 stop-bit.

The I-87H17W send a HART command to a device:

The “Byte Count” indicates the number of data bytes in the range 0 to 255.

Preamble	Delimiter	Address	Command	Byte Count	Data	Check byte
5 - 20 bytes	1 byte	1 bytes (short) 5 bytes (long)	1 byte	1 byte	0 - 255 bytes	1 byte

The device responds data to the I-87H17W:

The “Byte Count” indicates the number of response code and data bytes in the range 0 to 255.

Preamble	Delimiter	Address	Command	Byte Count	Response code	Data	Check byte
5 - 20 bytes	1 byte	1 bytes (short) 5 bytes (long)	1 byte	1 byte	2 bytes	0 - 253 bytes	1 byte

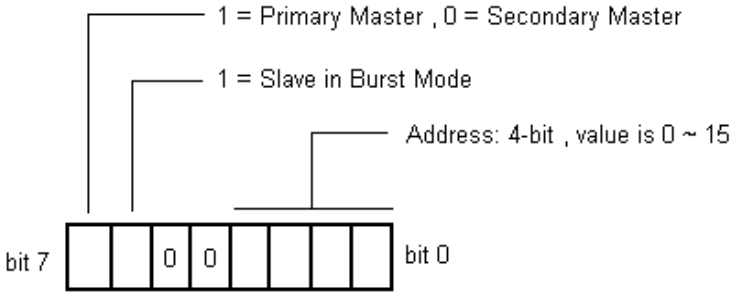
Preamble: 5 - 20 bytes
The value of each byte is “255” (i.e., 16#FF).

Delimiter: 1 byte

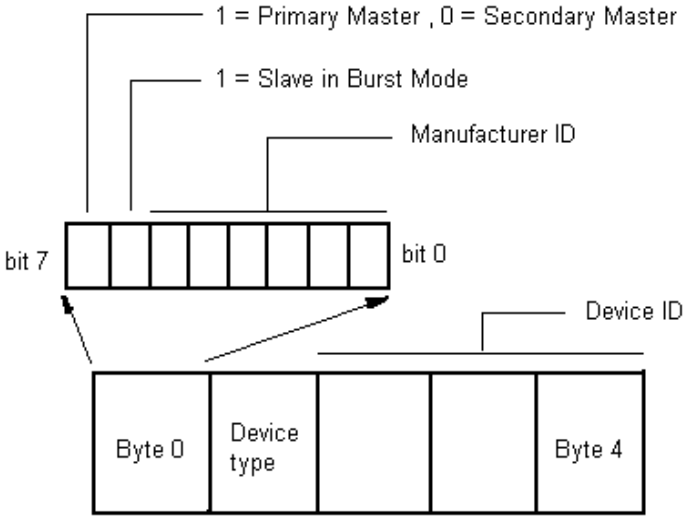
01 :		Short frame, Burst Frame.
02 :		Short frame, Master to Slave.
06 :		Short frame, Slave to Master.
129	(16#81):	Long frame, Burst Frame.
130	(16#82):	Long frame, Master to Slave.
134	(16#86):	Long frame, Slave to Master.

Address: 1 byte (Short frame) or 5 bytes (Long frame)

Short frame (1 byte):



Long frame (5 bytes):



Command: 1 byte (refer the manual of the HART device for its definition).

Byte Count: 1 byte.

Send (I-87H17W): "Byte Count" is the byte amount of the "Data". The value is 0 to 255.

Response (device): "Byte Count" is the byte amount of "Data" + "Response code". The value is 0 to 255.

Response code: 2 byte (refer the manual of the HART device for its definition).

Data: Refer the manual of the HART device for its definition.

Check Byte: 1 byte. The user can ignore the "Check Byte" in the Win-GRAF program because the I-87H17W module will automatically add (or remove) it when sending (or receiving) the HART frame.

23.3 Introduction of the Win-GRAF Demo Project

This section lists three Win-GRAF projects to show you the way to do HART communication between the Win-GRAF PAC and HART devices. The user can run the Win-GRAF Workbench and then click "File → Add Existing Project → From Zip..." to open the Win-GRAF project, which is stored in a shipping CD (CD-ROM: \Napdos\Win-GRAF\demo-project\).

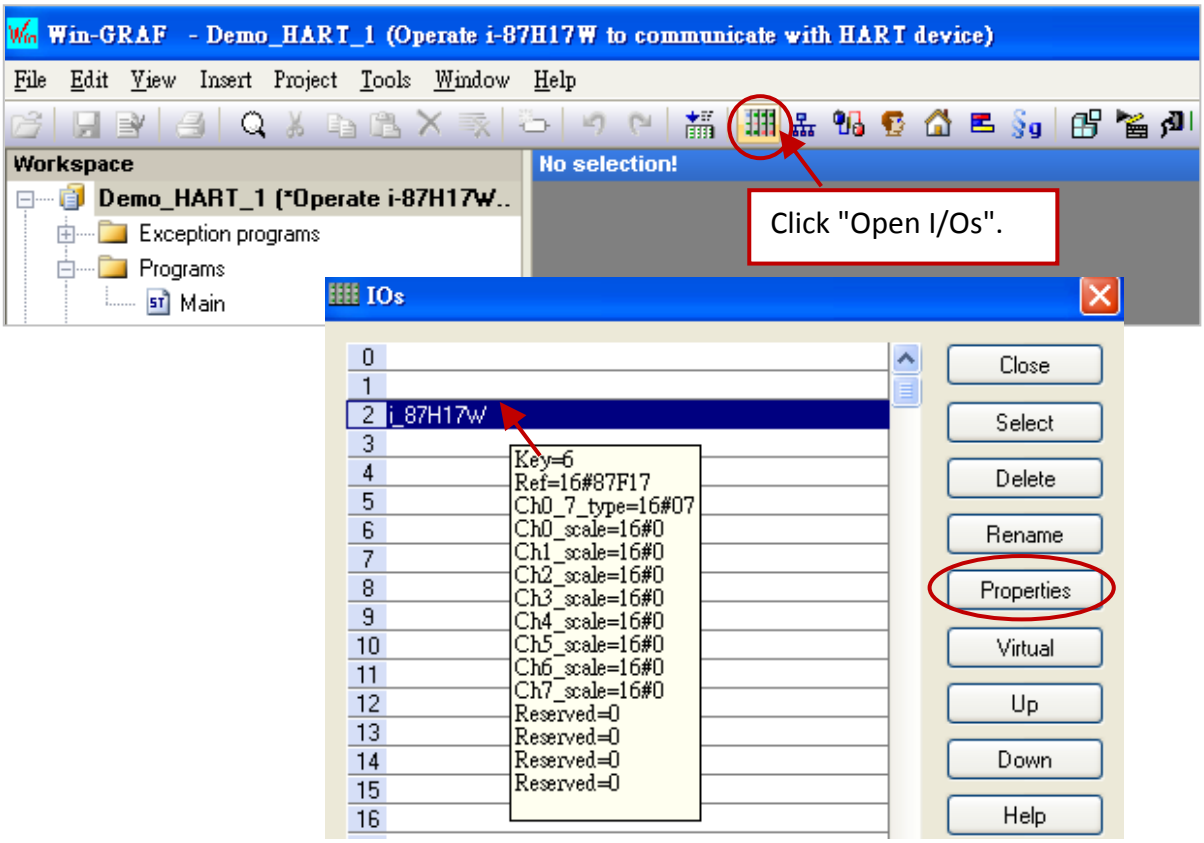
File Name	Description
Demo_HART_1.zip	Sending the HART frame manually from Ch-0 of the I-87H17W in the slot 2, and then receive the HART frame from the device.
Demo_HART_2.zip	Sending and receiving the HART frame every 5 seconds automatically and repeatedly from Ch-0, Ch-1 and Ch-2 of the I-87H17W in the slot 2.
Demo_HART_3.zip	(It's similar as Demo_HART_2) Using two I-87H17W in the slot 2 and slot 3 to send and receive the HART frame every 5 seconds automatically and repeatedly from Ch-0 to Ch-7 at the same time.

Note: Because all 8 channels of the I-87H17W are shared with one HART chip. Each I-87H17W can conduct only one channel for sending/receiving HART frame at a time, and then conduct the next channel.

23.3.1 I/O Board Setting and the HART Functions

I/O Board Setting

Both the "Demo_HART_1" and the "Demo_HART_2" projects require to use one I-87H17W on the PAC's Slot 2. Therefore, we need to enable this I/O function in the Win-GRAF "I/O Boards" window. Simply click the "Open I/Os" button and then add "i_87H17W" in the slot no. 2, click "Properties" can view more details about it (or see Chapter 4 – Linking I/O boards).



“I-87H17W” is an eight channels HART current input module (data type: REAL; input range: 4 - 20mA).

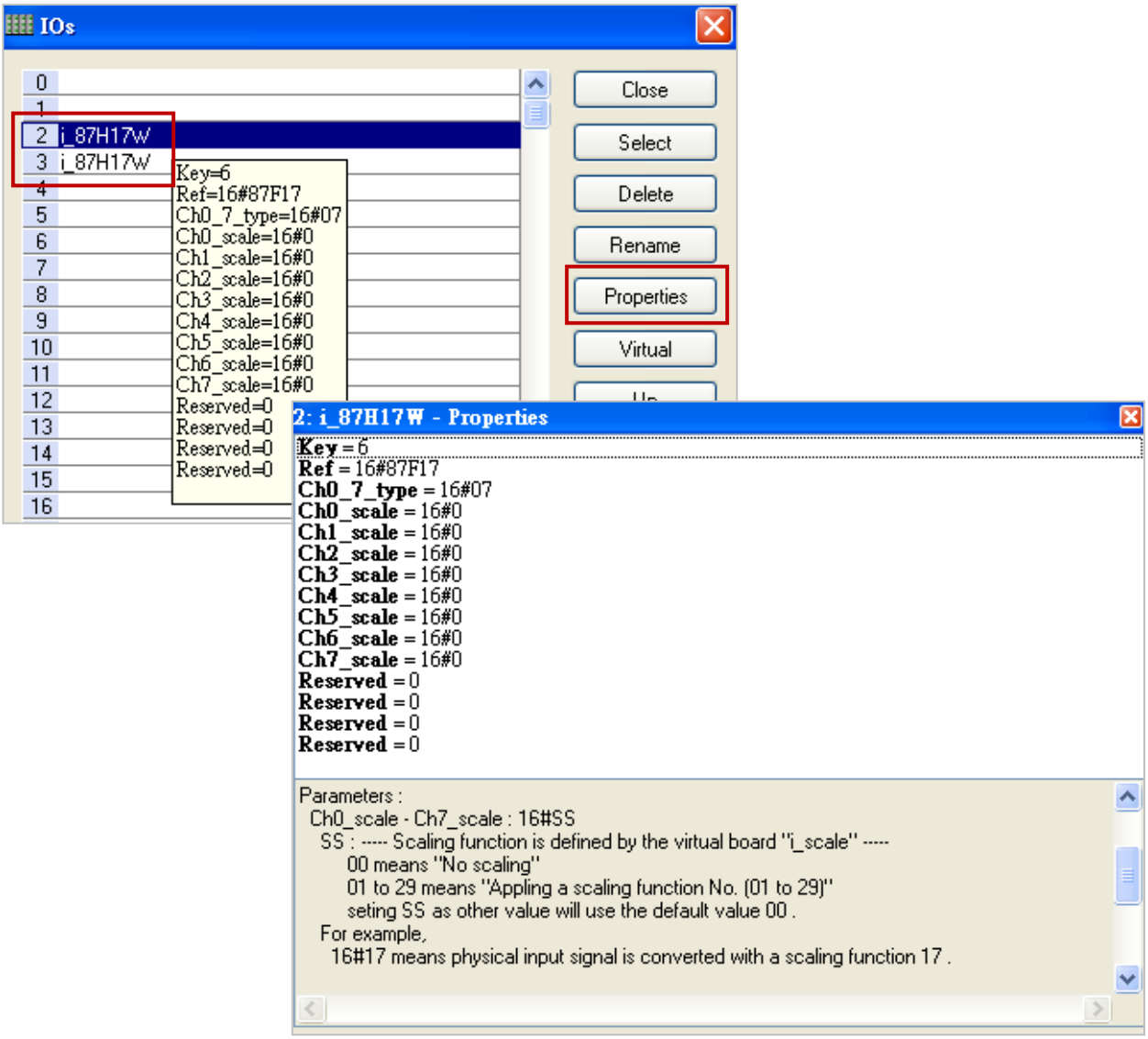
Parameters:
Ch0_scale ~ Ch7_scale : 16#SS

SS : Scaling function is defined by the "i_scale" I/O board (refer the Section 4.2).
00 means "No scaling".
01 to 29 means "Appling a scaling function No. (01 to 29)"
Setting SS as other value will use the default value 00.

For example, 16#17 means the physical input signal is converted with the scaling function 17.

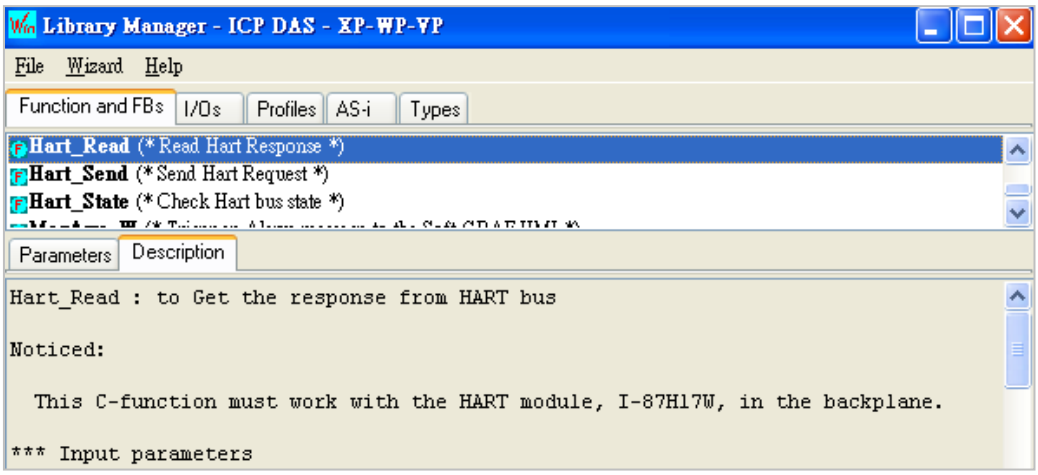
Note: If the Current value is greater than "9000.0", which means the HART device may be disconnected.

In the “Demo_HART_3” project, it need to enable two “i_87H17W” in the slot no. 2 and no.3.



The HART Functions

The user can use the "HART_Read", "HART_Send" and "HART_Status" functions in the ST or LD program. In addition, you can press "F1" or open "Library Manager" (refer Section 1.2.3) to view the descriptions of these functions.



Hart_Read: To get the response from HART Bus.

Note: This function must work with the HART module (i.e., I-87H17W), which is plugged in the Win-GRAF PAC.

Input Parameters:

gSlot: (Data type: DINT)
The specified I/O slot which the I-87H17W is plugged in. It used to read the response frame from the HART bus. (The value range is 0 ~ 7)

@gPreamble: (Data type: USINT) (Refer Section 23.2 – The format of HART frame.)
To get the Preamble number of HART response.

@gDelimiter: (Data type: USINT) To get the Delimiter of the HART response.
gAddress[]: (Data type: USINT) To get the Address of the HART response.
@gCommand: (Data type: USINT) To get the Command of the HART response.
gData[]: (Data type: USINT) To get the Data of the HART response.
@gDataLen: (Data type: USINT) To get the data length of the HART response.

Output Parameters:

Q : (Data type: DINT)

0 : Nothing happened.
1 : Read success.
-1 : Cannot find the HART module, I-87H17W, in the specified slot.
-2 : Invalid HART response.
-3 : Waiting HART response.
-4 : No HART request.
-5 : HART module I-87H17 is off line.

hart_Send: To send the request to HART bus.

Note: This function must work with the HART module (i.e., I-87H17W), which is plugged in the Win-GRAF PAC.

Input Parameters:

gSlot: (Data type: DINT)
The specified I/O slot which the I-87H17W is plugged in. It used to send the HART request. (The value range is 0 - 7.)

gChannel: (Data type: DINT)
The specified channel of I-87H17W to send HART request to HART device.
(The value range is 0 - 7.)

gPreamble: (Data type: USINT) (Refer Section 23.2 – The format of HART frame.)
The Preamble amount of HART request (The value range is 5 - 20.)

gDelimiter: (Data type: USINT)
The Delimiter of the HART request.

gAddress[]: (Data type: USINT)
The Address of the HART request.

gCommand: (Data type: USINT)
The Command of the HART request.

gData[]: (Data type: USINT)
The Data of the HART request.

gDataLen: (Data type: USINT)
The Data length (byte) of the HART request.

Output Parameters:

Q : (Data type: DINT)

0 : Nothing happened.
1 : Read success.
-1 : Cannot find the HART module, i-87H17W, in the specified slot.
-2 : The specified channel of I-87H17W is wrong.
-3 : The specified Preamble amount is wrong.
-4 : The sending error of the HART request, please check Delimiter and Address is correct.
-5 : HART bus is busy.
-6 : HART module I-87H17 is off line.

Hart_State: To Get HART bus state.

Note: This function must work with the HART module (i.e., I-87H17W), which is plugged in the Win-GRAF PAC.

Input Parameters:

gSlot: (Data type: DINT)

The specified I/O slot which the I-87H17W is plugged in. It used to get the status of the HART bus.

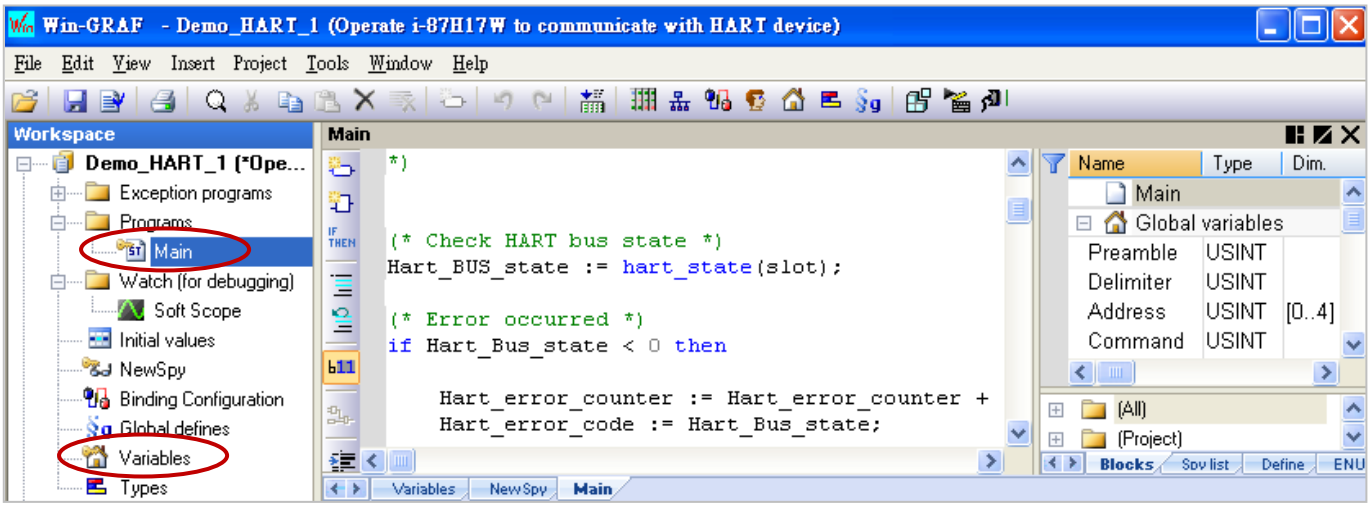
Output Parameters:

Q : (Data type: DINT)

0 : No any query.
1 : HART request is waiting to send to HART device.
2 : HART request is sent to HART device.
3 : Waiting the HART response.
4 : HART response is received.
-1 : Timeout.
-2 : The length of the HART response frame is too short.
(Refer Section 23.2 – The format of HART frame.)
-3 : The Delimiter of the HART response is wrong.
-4 : The Address (Master) of the HART response is wrong.
-5 : The Address (Burst) of the HART response is wrong.
-6 : The Command of the HART response is wrong.
-7 : The Checksum of the HART response is wrong.
-8 : HART response error.
-98 : There is some mismatch between the parameter of the HART response and the HART request.
-99 : Impossible error.
-100 : The error happened during sending the HART request to the HART device.
-101 : Cannot find HART module I-87H17W in the specified slot.
-102 : HART module I-87H17 is off line.

23.3.2 The Demo Project - “Demo_HART_1”

(You can see the feature description in the ST program below.)



Description of Variables:

Variables Name	Data Type	Dim.	Description
slot	DINT	-	Used to send the HART request. (The initial value of “slot” is set to “2”) (Refer Section 23.2 - The format of HART frame. Section 23.3.1 - "HART_Send" function)
channel	DINT	-	
Preamble	USINT	-	
Delimiter	USINT	-	
Address	USINT	[0..4]	
Command	USINT	-	
Data	USINT	[0..254]	
Datalen	USINT	-	
ret	DINT	-	Send/receive the status value of the HART frame.
Preamble_r	USINT	-	Used to receive the HART response. (Refer Section 23.2 - The format of HART frame. Section 23.3.1 - "HART_Send" function)
Delimiter_r	USINT	-	
Address_r	USINT	[0..4]	
Command_r	USINT	-	
Data_r	USINT	[0..254]	
Datalen_r	USINT	-	
Hart_BUS_state	DINT	-	To get the status of the HART bus.
Hart_error_counter	DINT	-	To calculate error times of the HART bus.
Hart_error_code	DINT	-	The error code of the HART bus.
hart_bus_err_Msg	STRING(255)	-	The error message of the HART bus.
hart_send_err_msg	STRING(255)	-	The error message when sending the HART frame.
hart_read_err_msg	STRING(255)	-	The error message when reading the HART frame.
send_HART_request	BOOL	-	Set it as “TRUE” to send the HART frame.
read_HART_request	BOOL	-	Set it as “TRUE” to read the HART frame.

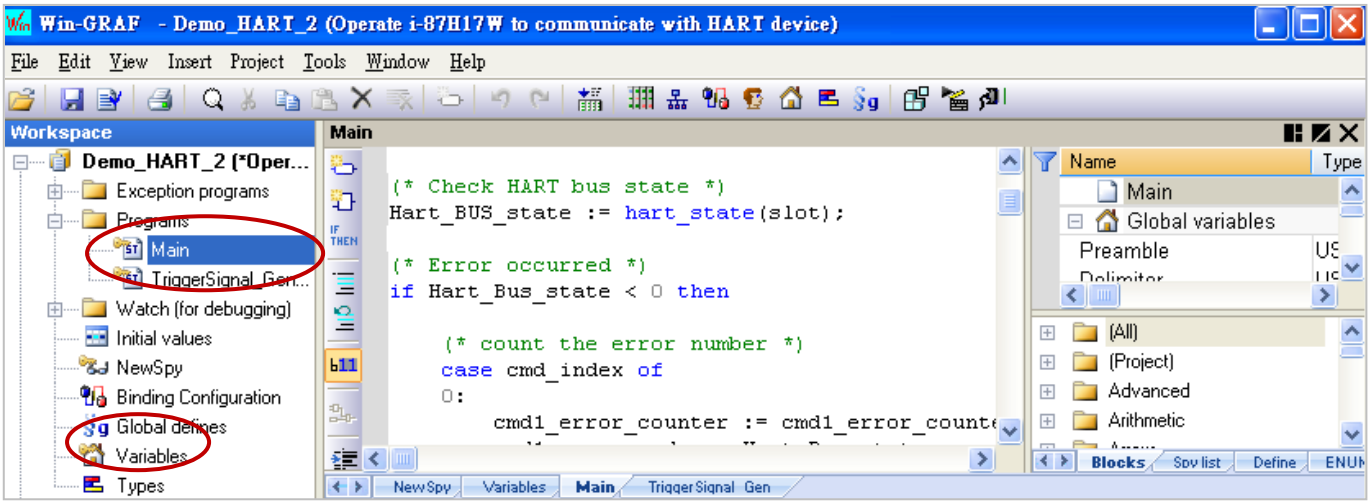
ST Program (Main)

```
(*  This demo project is to show how to use one I-87H17W(HART Master) to communicate
    manually with one HART device.

   Hardware Environment:

   1. Plug i-87H17W in the slot 2 of Win-GRAF PAC.
   2. Connect HART device to the channel 0 of i-87H17W  *)

(* Check HART bus state *)
Hart_BUS_state := hart_state(slot);

(* Error message and error times of the HART Bus *)
if Hart_Bus_state < 0 then

   Hart_error_counter := Hart_error_counter + 1;
   Hart_error_code := Hart_Bus_state;

   case Hart_error_code of
     -1 : hart_bus_err_Msg := 'Error: Timeout' ;
     -2 : hart_bus_err_Msg := 'Error: Read_data_too_short error' ;
     -3 : hart_bus_err_Msg := 'Error: Response Delimiter error' ;
     -4 : hart_bus_err_Msg := 'Error: Response addr_master error' ;
     -5 : hart_bus_err_Msg := 'Error: Response addr_burst error' ;
     -6 : hart_bus_err_Msg := 'Error: Response recv_command error' ;
     -7 : hart_bus_err_Msg := 'Error: Response checksum error' ;
     -8 : hart_bus_err_Msg := 'Error: response error' ;
    -98 : hart_bus_err_Msg := 'Error: para_mismatch error' ;
    -99 : hart_bus_err_Msg := 'Error: impossible error' ;
    -100: hart_bus_err_Msg := 'Error: Sending HART request error' ;
    -101: hart_bus_err_Msg := 'Error: cannot find the HART moudule, i-87H17W' ;
    -102: hart_bus_err_Msg := 'Error: i-87H17W is off-line';
   end_case;

end_if;


if Hart_BUS_state = 0 and Send_HART_request then
(* HART bus is free to send *)

   (* Build HART command *)
   slot := 2;
   channel := 0;
   Preamble := 5;
   Delimiter := 16#82;
   Address[0] := 16#96;
   Address[1] := 16#85;
```

```
   Address[2] := 16#0B;
   Address[3] := 16#0A;
   Address[4] := 16#42;
   Command := 16#03;
   Datalen := 0;

   ret := hart_send(slot, channel, Preamble, Delimiter, Address, Command, Data, Datalen);

   case ret of
    1: Send_HART_request := false;
       hart_send_err_msg := 'Send success';
   -1: hart_send_err_msg := 'Send Error: cannot find the HART moudule, i-87H17W';
   -2: hart_send_err_msg := 'Send Error: specified the wrong channel to send hart cmd';
   -3: hart_send_err_msg := 'Send Error: specified the wrong preamble number';
   -4: hart_send_err_msg := 'Send Error: wrong HART cmd, please check delimiter and
       address is correct';
   -5: hart_send_err_msg := 'Send Error: HART bus is busy';
   -6: hart_send_err_msg := 'Send Error: i-87H17W is off-line';
   end_case;

end_if;

(* HART response is coming *)
if Hart_BUS_state = 4 then

   ret := hart_read(slot, Preamle_r, Delimiter_r, Address_r, Command_r, Data_r, Datalen_r);

   case ret of
    1: hart_read_err_msg := 'Read success';
       read_HART_request := true;
   -1: hart_read_err_msg := 'Read Error: cannot find the HART moudule, i-87H17W';
   -2: hart_read_err_msg := 'Read Error: invalid HART response';
   -3: hart_read_err_msg := 'Read Error: waiting response';
   -4: hart_read_err_msg := 'Read Error: No Request';
   -5: hart_read_err_msg := 'Read Error: i-87H17W is off-line';
   end_case;

end_if;

(* HART response processing *)
if read_HART_request then
   read_HART_request := false;

   (* ToDo: put your code here to handle the HART response *)


end_if;
```

23.3.3 The Demo Project - “Demo_HART_2”

(You can see the feature description in the ST program below.)



Description of Variables:

Variables Name	Data Type	Dim.	Description
slot	DINT	-	Used to send the HART request. (The initial value of “slot” is set to “2”) (Refer Section 23.2 - The format of HART frame. Section 23.3.1 - "HART_Send" function)
channel	DINT	-	
Preamble	USINT	-	
Delimiter	USINT	-	
Address	USINT	[0..4]	
Command	USINT	-	
Data	USINT	[0..254]	
Datalen	USINT	-	
ret	DINT	-	Send/receive the status value of the HART frame.
Preamble_r	USINT	-	Used to receive the HART response. (Refer Section 23.2 - The format of HART frame. Section 23.3.1 - "HART_Send" function)
Delimiter_r	USINT	-	
Address_r	USINT	[0..4]	
Command_r	USINT	-	
Data_r	USINT	[0..254]	
Datalen_r	USINT	-	
Hart_BUS_state	DINT	-	To get the status of the HART bus.
hart_bus_err_Msg	STRING(255)	-	The error message of the HART bus.
hart_send_err_msg	STRING(255)	-	The error message when sending the HART frame.
hart_read_err_msg	STRING(255)	-	The error message when reading the HART frame.
send_HART_request	BOOL	-	Set it as “TRUE” to send the HART frame.
read_HART_request	BOOL	-	Set it as “TRUE” to read the HART frame.
cmd_index	DINT	-	The index value to process the command.
cmd_max_num	DINT	-	The number of sending commands. (Init. = 3)

Variables Name	Data Type	Dim.	Description
cmd1_response_counter	DINT	-	The number of response times. (From Ch-0 / Ch-1/ Ch-2)
cmd2_response_counter	DINT	-	
cmd3_response_counter	DINT	-	
cmd1_error_counter	DINT	-	The number of error times on the HART bus. (From Ch-0 / Ch-1/ Ch-2)
cmd2_error_counter	DINT	-	
cmd3_error_counter	DINT	-	
cmd1_error_code	DINT	-	The error code of the HART bus. (From Ch-0 / Ch-1/ Ch-2)
cmd2_error_code	DINT	-	
cmd3_error_code	DINT	-	
trigger_Timer	TIME	-	Start ticking.
trigger_interval	TIME	-	The interval time for sending commands. (Init = T#5s)
INIT	BOOL	-	Initialize (Init. = TRUE).
next_index	DINT	-	The index for the response data.
cmd1_Data	REAL	[0..3]	Used to save the response data. (From Ch-0 / Ch-1/ Ch-2)
cmd2_Data	REAL	[0..3]	
cmd3_Data	REAL	[0..3]	

<u>ST Program</u> (Main)

```
(*  This demo project is to show how to use one I-87H17W(HART Master) to communicate
    automatically with HART devices that is connected with different channels of I-87H17W.

  Hardware Environment:

   1. Plug I-87H17W in the slot 2 of Win-GRAF PAC.
   2. Connect three HART devices to the channel 0 ~ 2 of i-87H17W in the slot 2 of Win-GRAF PAC
    separeately.
*)

 (* Check HART bus state *)
Hart_BUS_state := hart_state(slot);

(* Error occurred *)
if Hart_Bus_state < 0 then

   (* Count the error number *)
   case cmd_index of
   0:
      cmd1_error_counter := cmd1_error_counter + 1;
      cmd1_error_code := Hart_Bus_state;
   1:
      cmd2_error_counter := cmd2_error_counter + 1;
      cmd2_error_code := Hart_Bus_state;
```

```
   2:
      cmd3_error_counter := cmd3_error_counter + 1;
      cmd3_error_code := Hart_Bus_state;
   end_case;

   (* Try to send the next HART request *)
   cmd_index := cmd_index + 1;

   if cmd_index = cmd_max_num then
      cmd_index := 0;
   end_if;


   case Hart_Bus_state of
     -1 : hart_bus_err_Msg := 'Error: Timeout' ;
     -2 : hart_bus_err_Msg := 'Error: Read_data_too_short error' ;
     -3 : hart_bus_err_Msg := 'Error: Response Delimiter error' ;
     -4 : hart_bus_err_Msg := 'Error: Response addr_master error' ;
     -5 : hart_bus_err_Msg := 'Error: Response addr_burst error' ;
     -6 : hart_bus_err_Msg := 'Error: Response recv_command error' ;
     -7 : hart_bus_err_Msg := 'Error: Response checksum error' ;
     -8 : hart_bus_err_Msg := 'Error: response error' ;
    -98 : hart_bus_err_Msg := 'Error: para_mismatch error' ;
    -99 : hart_bus_err_Msg := 'Error: impossible error' ;
    -100: hart_bus_err_Msg := 'Error: Sending HART request error' ;
    -101: hart_bus_err_Msg := 'Error: cannot find the HART moudule, i-87H17W' ;
    -102: hart_bus_err_Msg := 'Error: i-87H17W is off-line';
   end_case;

end_if;


if Hart_BUS_state = 0 and Send_HART_request then
(* HART bus is free to send *)

   (* Build HART command *)
   case cmd_index of
   0:
      slot := 2;
      channel := 0;
      Preamble := 5;
      Delimiter := 16#82;
      Address[0] := 16#96;
      Address[1] := 16#85;
      Address[2] := 16#0B;
      Address[3] := 16#0A;
      Address[4] := 16#42;
      Command := 16#03;
      Datalen := 16#00;
```

```
   1:
      slot := 2;
      channel := 1;
      Preamble := 5;
      Delimiter := 16#82;
      Address[0] := 16#96;
      Address[1] := 16#85;
      Address[2] := 16#0B;
      Address[3] := 16#0A;
      Address[4] := 16#42;
      Command := 16#03;
      Datalen := 16#00;
    2:
      slot := 2;
      channel := 2;
      Preamble := 5;
      Delimiter := 16#82;
      Address[0] := 16#96;
      Address[1] := 16#85;
      Address[2] := 16#0B;
      Address[3] := 16#0A;
      Address[4] := 16#42;
      Command := 16#03;
      Datalen := 16#00;
   end_case;

   ret := hart_send(slot, channel, Preamble, Delimiter, Address, Command, Data, Datalen);

   case ret of
    1: Send_HART_request := false;
      hart_send_err_msg := 'Send success';
   -1: hart_send_err_msg := 'Send Error: cannot find the HART moudule, i-87H17W';
   -2: hart_send_err_msg := 'Send Error: specified the wrong channel to send hart cmd';
   -3: hart_send_err_msg := 'Send Error: specified the wrong preamble number';
   -4: hart_send_err_msg := 'Send Error: wrong HART cmd, please check delimiter and address is
         correct';
   -5: hart_send_err_msg := 'Send Error: HART bus is busy';
   -6: hart_send_err_msg := 'Send Error: i-87H17W is off-line';
   end_case;

end_if;

(* HART response is coming *)
if Hart_BUS_state = 4 then

   ret := hart_read(slot, Preamle_r, Delimiter_r, Address_r, Command_r, Data_r, Datalen_r);

   case ret of
    1: hart_read_err_msg := 'Read success';
      read_HART_request := true;
```

```
  -1: hart_read_err_msg := 'Read Error: cannot find the HART moudule, i-87H17W';
  -2: hart_read_err_msg := 'Read Error: invalid HART response';
  -3: hart_read_err_msg := 'Read Error: waiting response';
  -4: hart_read_err_msg := 'Read Error: No Request';
  -5: hart_read_err_msg := 'Read Error: i-87H17W is off-line';
  end_case;

end_if;

(* HART response processing *)
if read_HART_request then
  read_HART_request := false;

  (*--- Please use your own <Data> definition for your own application ---
    The data of the HART response assume as 00 00 41 A0 FF 3E 0C 3E C5 37 48 20 41 C8 3F 22 39
    42 C9 8E D1
    The HART answer is from a device, which <Data> in big endian has the following meaning.
    41 A0 FF 3E = 41A0FF3E   < convert these 4 bytes (IEEE-754)  to one REAL> :  20.1246 mA
    3E C5 37 48 = 3EC53748   < convert these 4 bytes (IEEE-754)  to one REAL> :  0.385187
    41 C8 3F 22 = 41C83F22   < convert these 4 bytes (IEEE-754)  to one REAL> :  25.0308
    42 C9 8E D1 = 42C98ED1   < convert these 4 bytes (IEEE-754)  to one REAL> :  100.779      *)
  case cmd_index of
  0:
    cmd1_response_counter := cmd1_response_counter + 1;

    next_index := SerializeIn(Data_r, cmd1_Data[0], 2, true);
    next_index := next_index + 1;
    next_index := SerializeIn(Data_r, cmd1_Data[1], next_index, true);
    next_index := next_index + 1;
    next_index := SerializeIn(Data_r, cmd1_Data[2], next_index, true);
    next_index := next_index + 1;
    next_index := SerializeIn(Data_r, cmd1_Data[3], next_index, true);

  1:
    cmd2_response_counter := cmd2_response_counter + 1;

    next_index := SerializeIn(Data_r, cmd2_Data[0], 2, true);
    next_index := next_index + 1;
    next_index := SerializeIn(Data_r, cmd2_Data[1], next_index, true);
    next_index := next_index + 1;
    next_index := SerializeIn(Data_r, cmd2_Data[2], next_index, true);
    next_index := next_index + 1;
    next_index := SerializeIn(Data_r, cmd2_Data[3], next_index, true);
  2:
    cmd3_response_counter := cmd3_response_counter + 1;

    next_index := SerializeIn(Data_r, cmd3_Data[0], 2, true);
    next_index := next_index + 1;
```

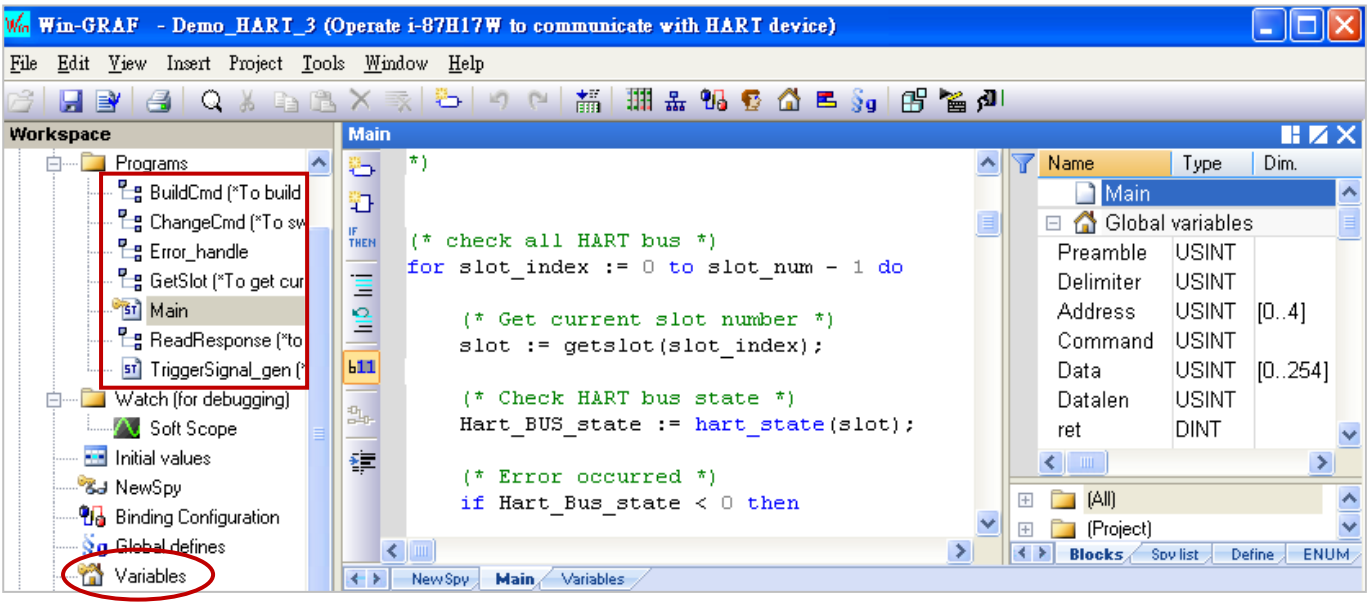
```
        next_index := SerializeIn(Data_r, cmd3_Data[1], next_index, true);
        next_index := next_index + 1;
        next_index := SerializeIn(Data_r, cmd3_Data[2], next_index, true);
        next_index := next_index + 1;
        next_index := SerializeIn(Data_r, cmd3_Data[3], next_index, true);
    end_case;

    (* Try to send the next HART request *)
    cmd_index := cmd_index + 1;
    Send_HART_request := true;

    if cmd_index = cmd_max_num then
        cmd_index := 0;
    end_if;

end_if;
```

<u>ST Program</u> (TriggerSignal)

```
(*
  To start the timer for generating the trigger signal to send HART request
*)

if INIT then
    INIT := false;
    TStart(trigger_timer);
    send_HART_request := true;
end_if;


if trigger_Timer > trigger_interval then

    trigger_Timer := T#0s;
    send_HART_request := true;

end_if;
```

23.3.4 The Demo Project - “Demo_HART_3”

(You can see the feature description in the ST program below.)



Description of Variables:

This demo project is similar as the **“Demo_HART_2”** project. The user can refer its variable descriptions or open the “Demo_HART_3” project to view all used variables in the Win-GRAF “Variables” window .

ST Program (Main)

There are several subroutine used in this project, the user can open them to view the program code. The following description is focused on the "Main" program.

```
(*
   This demo project is to show how to use two I-87H17W (HART Master) to communicate
   automatically with HART devices that is connected with different channels of I-87H17W.

  Hardware Environment:

   1. Plug one I-87H17W in the slot 2 of Win-GRAF PAC
   2. Plug the other one in the slot 3 of Win-GRAF PAC.
   3. Connect 8 HART devices to the channel 0 ~ 7 of I-87H17W in the slot 2 of Win-GRAF PAC
      separeately.
   4. Connect 8 HART devices to the channel 0 ~ 7 of I-87H17W in the slot 3 of Win-GRAF PAC
      separeately.  *)

(* Check all HART bus *)
 for slot_index := 0 to slot_num - 1 do

   (* Get current slot number *)
   slot := getslot(slot_index);

   (* Check HART bus state *)
   Hart_BUS_state := hart_state(slot);

  (* Error occurred *)
   if Hart_Bus_state < 0 then
```

```
    (* Count the error number *)
    Error_handle(any_to_byte(slot_index));

    (* Try to send the next HART request *)
    ChangeCmd(any_to_byte(slot_index));

    case Hart_Bus_state of
     -1 : hart_bus_err_Msg := 'Error: Timeout' ;
     -2 : hart_bus_err_Msg := 'Error: Read_data_too_short error' ;
     -3 : hart_bus_err_Msg := 'Error: Response Delimiter error' ;
     -4 : hart_bus_err_Msg := 'Error: Response addr_master error' ;
     -5 : hart_bus_err_Msg := 'Error: Response addr_burst error' ;
     -6 : hart_bus_err_Msg := 'Error: Response recv_command error' ;
     -7 : hart_bus_err_Msg := 'Error: Response checksum error' ;
     -8 : hart_bus_err_Msg := 'Error: response error' ;
    -98 : hart_bus_err_Msg := 'Error: para_mismatch error' ;
    -99 : hart_bus_err_Msg := 'Error: impossible error' ;
    -100: hart_bus_err_Msg := 'Error: Sending HART request error' ;
    -101: hart_bus_err_Msg := 'Error: cannot find the HART moudule, i-87H17W' ;
    -102: hart_bus_err_Msg := 'Error: i-87H17W is off-line';
    end_case;

  end_if;

  if Hart_BUS_state = 0 and Send_HART_request[slot_index] then
  (* HART bus is free to send *)

    (* Build HART command *)
    buildCmd(any_to_byte(slot_index));

    ret := hart_send(slot, channel, Preamble, Delimiter, Address, Command, Data, Datalen);

    case ret of
     1: Send_HART_request[slot_index] := false;
       hart_send_err_msg := 'Send success';
    -1: hart_send_err_msg := 'Send Error: cannot find the HART moudule, i-87H17W';
    -2: hart_send_err_msg := 'Send Error: specified the wrong channel to send hart cmd';
    -3: hart_send_err_msg := 'Send Error: specified the wrong preamble number';
    -4: hart_send_err_msg := 'Send Error: wrong HART cmd, please check delimiter and address
        is correct';
    -5: hart_send_err_msg := 'Send Error: HART bus is busy';
    -6: hart_send_err_msg := 'Send Error: i-87H17W is off-line';
    end_case;

  end_if;

(* HART response is coming *)
  if Hart_BUS_state = 4 then
    ret := hart_read(slot, Preamle_r, Delimiter_r, Address_r, Command_r, Data_r, Datalen_r);
```

```
      case ret of
       1: hart_read_err_msg := 'Read success';
         read_HART_request := true;
      -1: hart_read_err_msg := 'Read Error: cannot find the HART moudule, i-87H17W';
      -2: hart_read_err_msg := 'Read Error: invalid HART response';
      -3: hart_read_err_msg := 'Read Error: waiting response';
      -4: hart_read_err_msg := 'Read Error: No Request';
      -5: hart_read_err_msg := 'Read Error: i-87H17W is off-line';
      end_case;

    end_if;

    (* HART response processing *)
    if read_HART_request then
      read_HART_request := false;

      ReadResponse(any_to_byte(slot_index));
      ChangeCmd(any_to_byte(slot_index));
    end_if;
  end_for;
```

23.4 Test the Demo Program

Feature Descriptions:

File Name	Description
Demo_HART_1.zip	Sending the HART frame manually from Ch-0 of the I-87H17W in the slot 2, and then receive the HART frame from the device.
Demo_HART_2.zip	Sending and receiving the HART frame every 5 seconds automatically and repeatedly from Ch-0, Ch-1 and Ch-2 of the I-87H17W in the slot 2.
Demo_HART_3.zip	(It's similar as Demo_HART_2) Using two I-87H17W in the slot 2 and slot 3 to send and receive the HART frame every 5 seconds automatically and repeatedly from Ch-0 to Ch-7 at the same time.

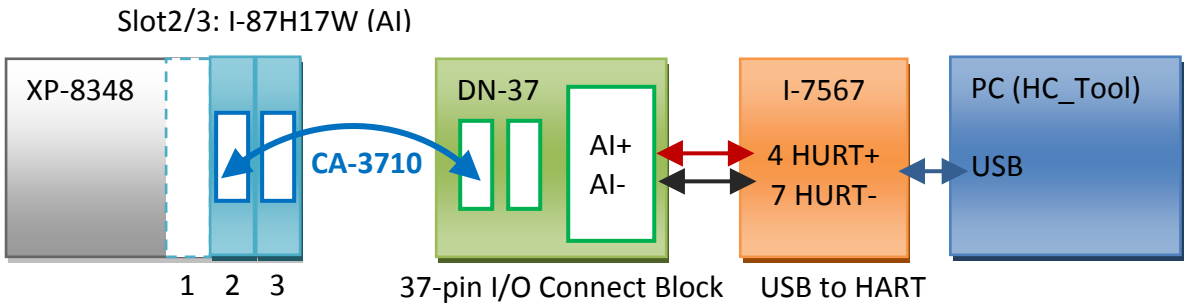
Note: Because all 8 channels of the I-87H17W are shared with one HART chip. Each I-87H17W can conduct only one channel for sending/receiving HART frame at a time, and then conduct the next channel.

23.4.1 Testing Environment and the "HC_Tool" Setting

Before testing, make sure the I-87H17W module is plugged in the slot2 (or slot3) of the Win-GRAF PAC, and the needed I/O channels (Ch-0 ~ Ch-7) are linked to HART devices (refer Section 23.1). Then, power on the Win-GRAF PAC and download the project to it.

The following will describe the test way for these Win-GRAF demo programs one by one. Here, we use the "HC_Tool" utility to simulate a HART device to conduct the HART response. The user can download "HC_Tool" and its user manual on the I-7567 webpage (as the list below).

Refer the following product page (Pin Assignment or Wire Connection) to know the way of wiring.

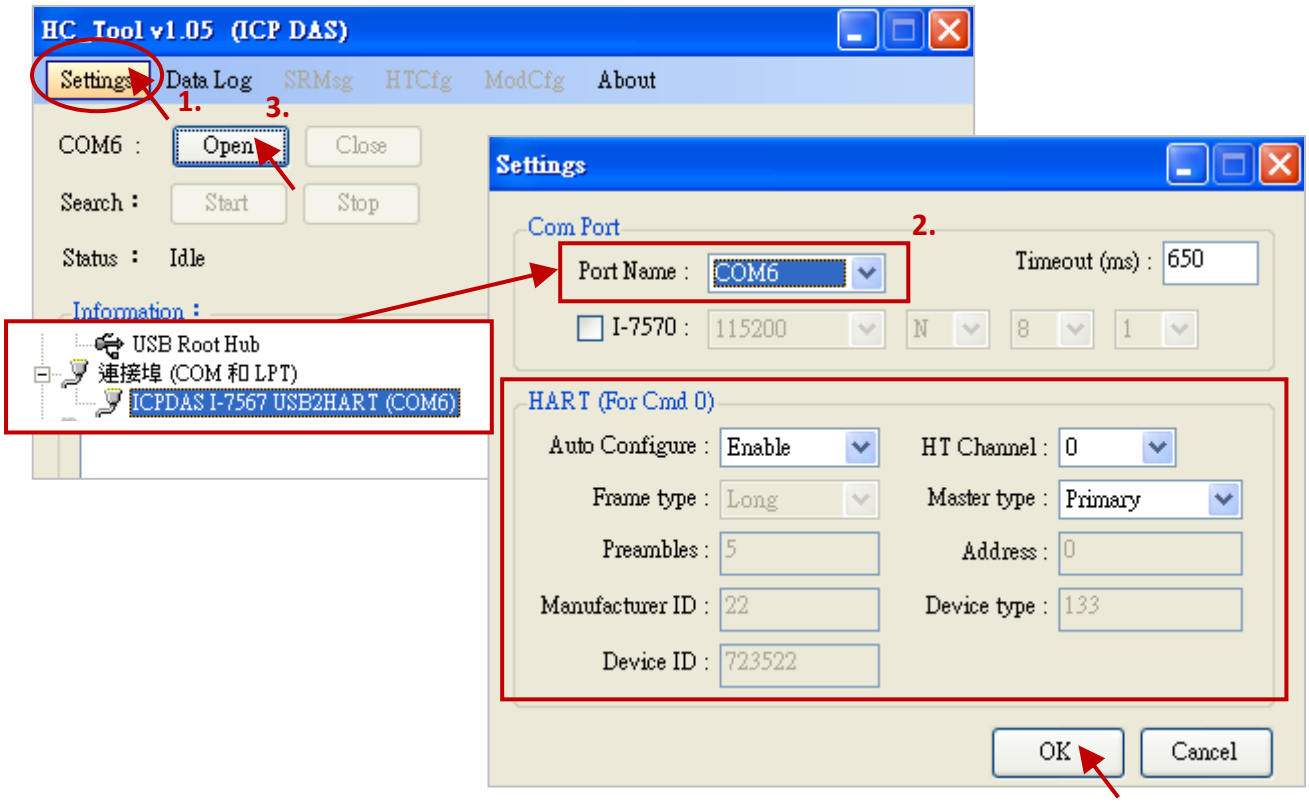


Testing Enviorment:

Device	Model	Quantity	Note
Win-GRAF PAC	XP-8348-CE6	1	-
HART AI Module	I-87H17W	2	The "Demo_HART_1" and "Demo_HART_2" projects require only one I-87H17W.
USB to HART Converter	I-7567	1	To download the USB driver and the "HC_Tool" utility.
Website: www.icpdas.com > Product >Industrial Communication > Fieldbus Solutions > HART Series			
I/O Connector Block	DN-37	1	Includes one CA-3710 cable.
Website: www.icpdas.com > Product > PC based I/O Board > PCI Bus I/O Boards > Daughter Boards			

The HC_Tool Setting:

Run "HC_Tool.exe" and then click "Setting" to select the COM Port no. which your PC used (see "Device Manager"). You can keep the settings like the figure below (or see I-7567 manual) and click "OK", and then click "Open" to open this COM Port.

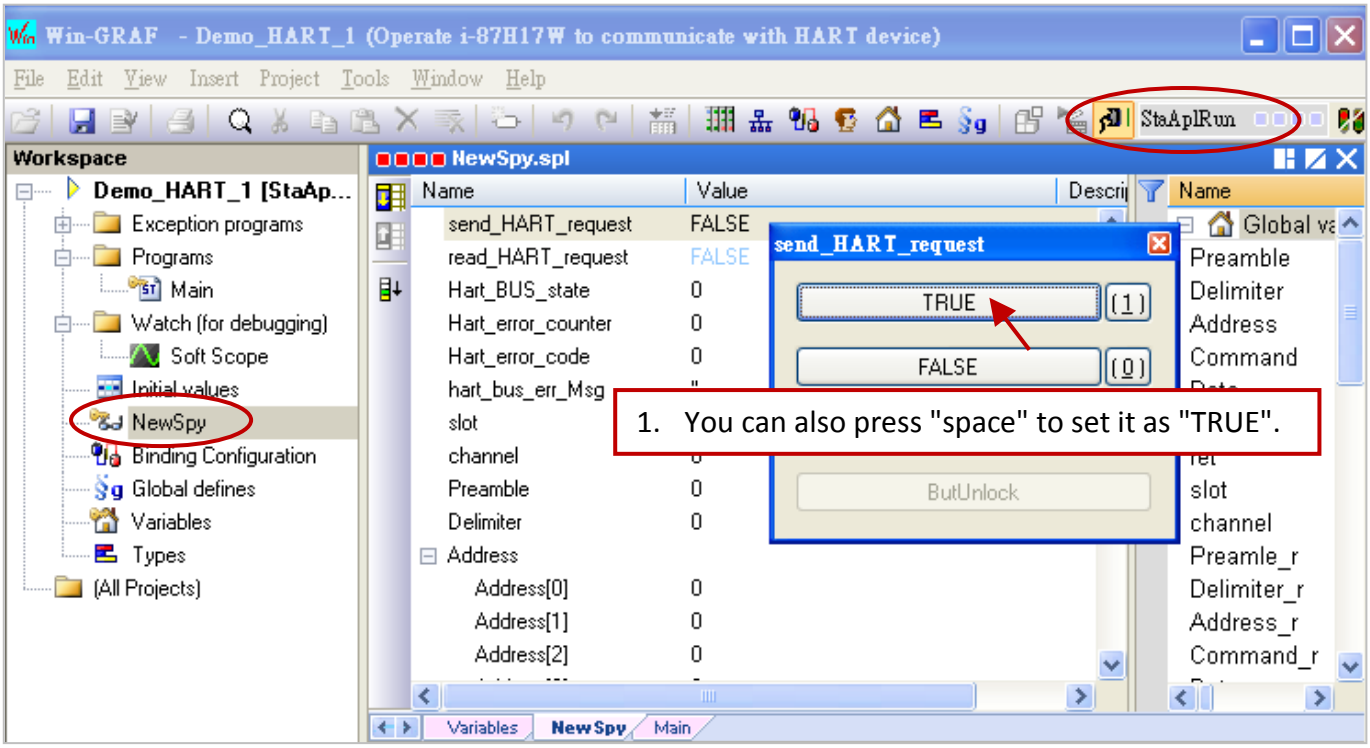


23.4.2 Test the "Demo_HART_1", "Demo_HART_2" and "Demo_HART_3" Projects

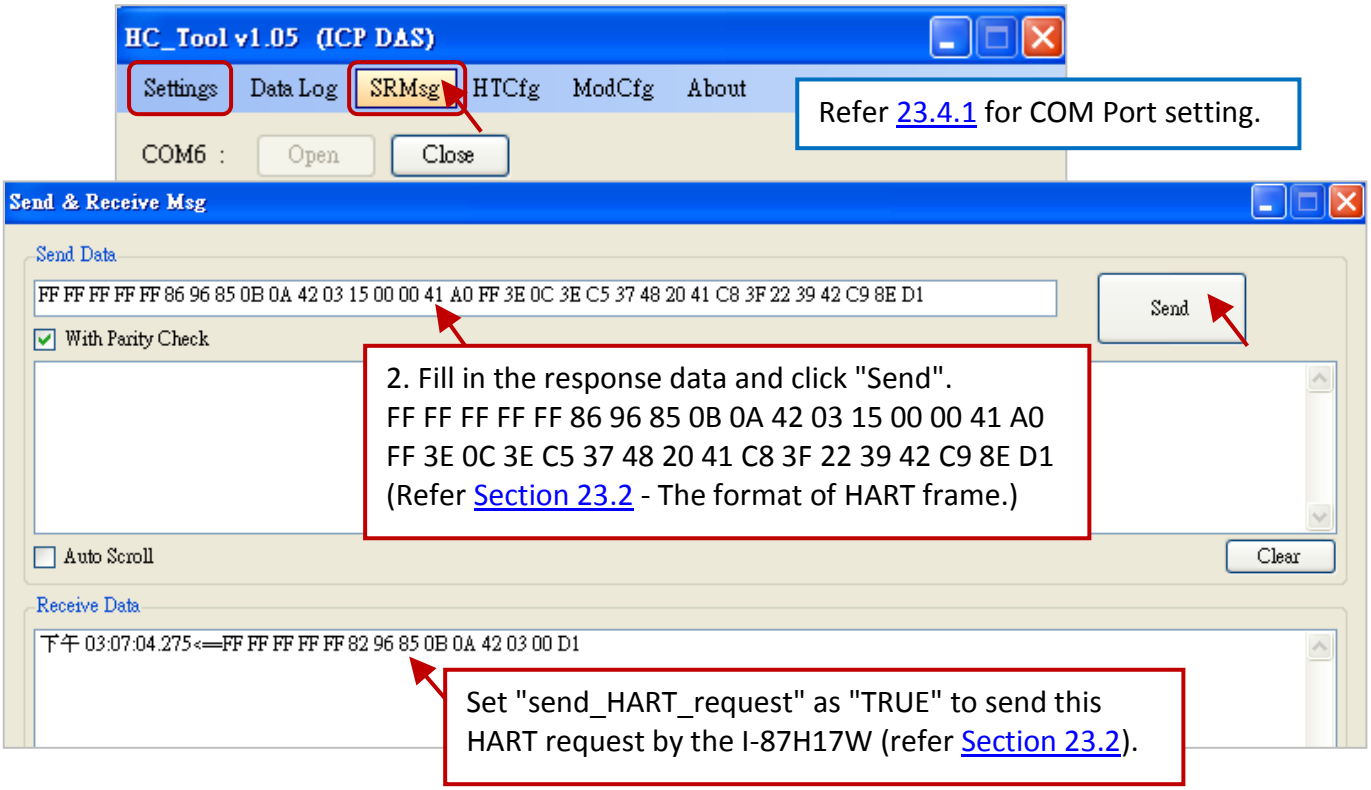
Feature Descriptions: (Program- Demo_HART_1 ; Wiring – See Section 23.4.1)

Sending the HART frame manually from Ch-0 of the I-87H17W in the slot 2, and then receive the HART frame from the device.

After downloading this project, open the spy list (NewSpy) and set the "send_HART_request" variable as "TRUE" to send the HART request.

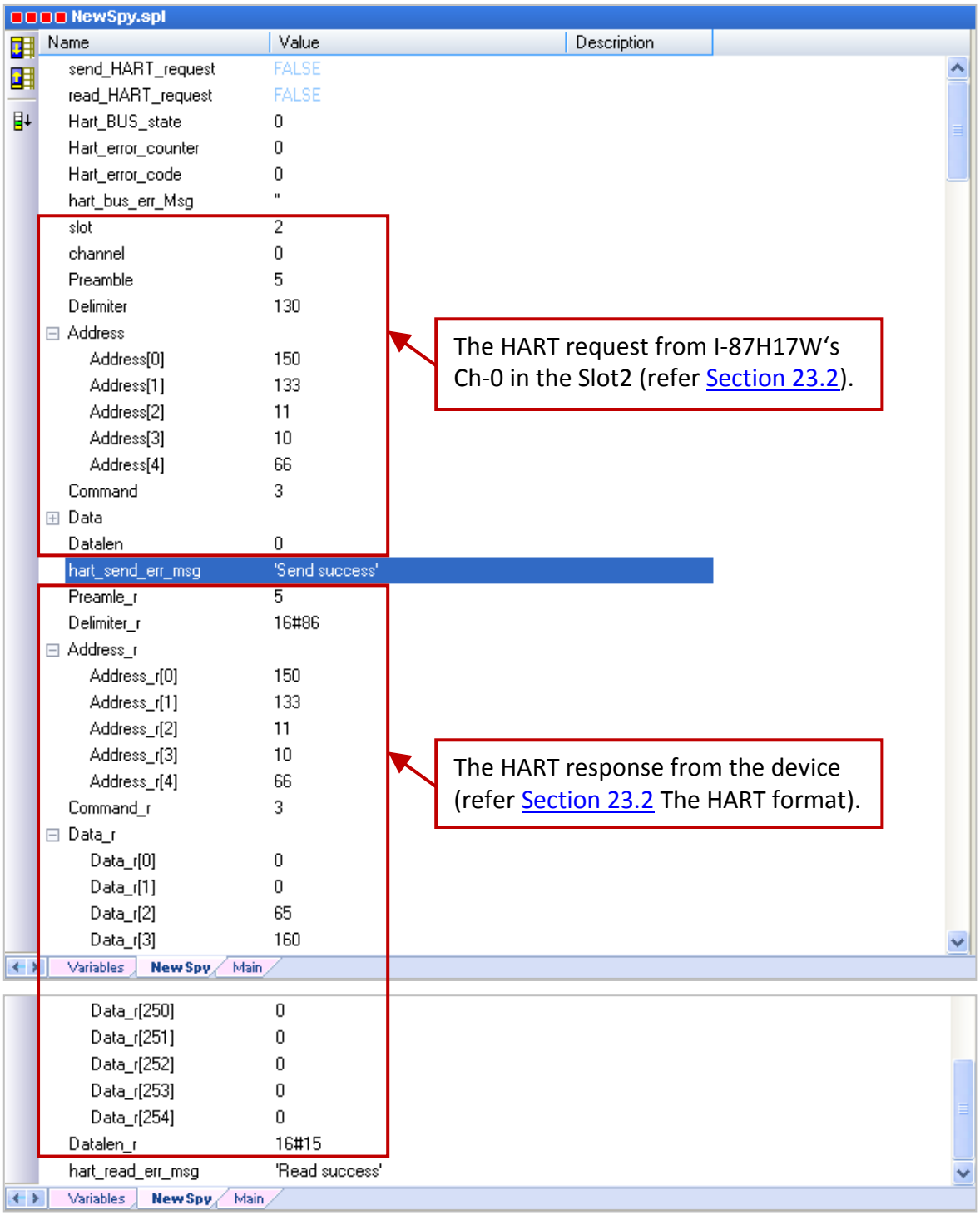


In the "HC_Tool", click "SRMsg" to open "Send & Receive Msg" window, fill in the response data and click "Send".



Preamble	Delimiter	Address	Command	Byte Count	Data
The HART request from the I-87H17W (Hex.)					
FF FF FF FF FF	FF FF FF FF FF	FF FF FF FF FF	FF FF FF FF FF	FF FF FF FF FF	FF FF FF FF FF
The HART response from the device (Hex.)					
FF FF FF FF FF	86	(long) 96 85 0B 0A 42	03	$15_{(16)}$	00 00 41 A0 FF 3E 0C 3E C5 37 48 20 41 C8 3F 22 39 42 C9 8E D1

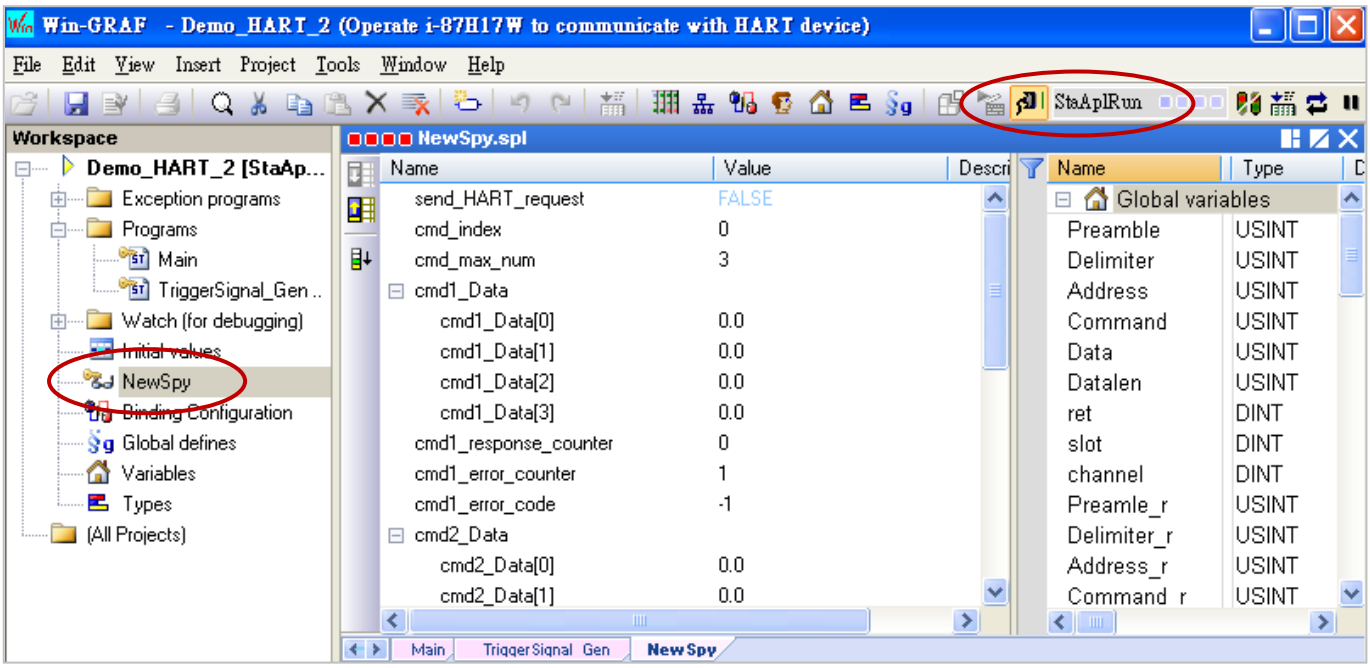
If the process is successful, you will see the screen like below.



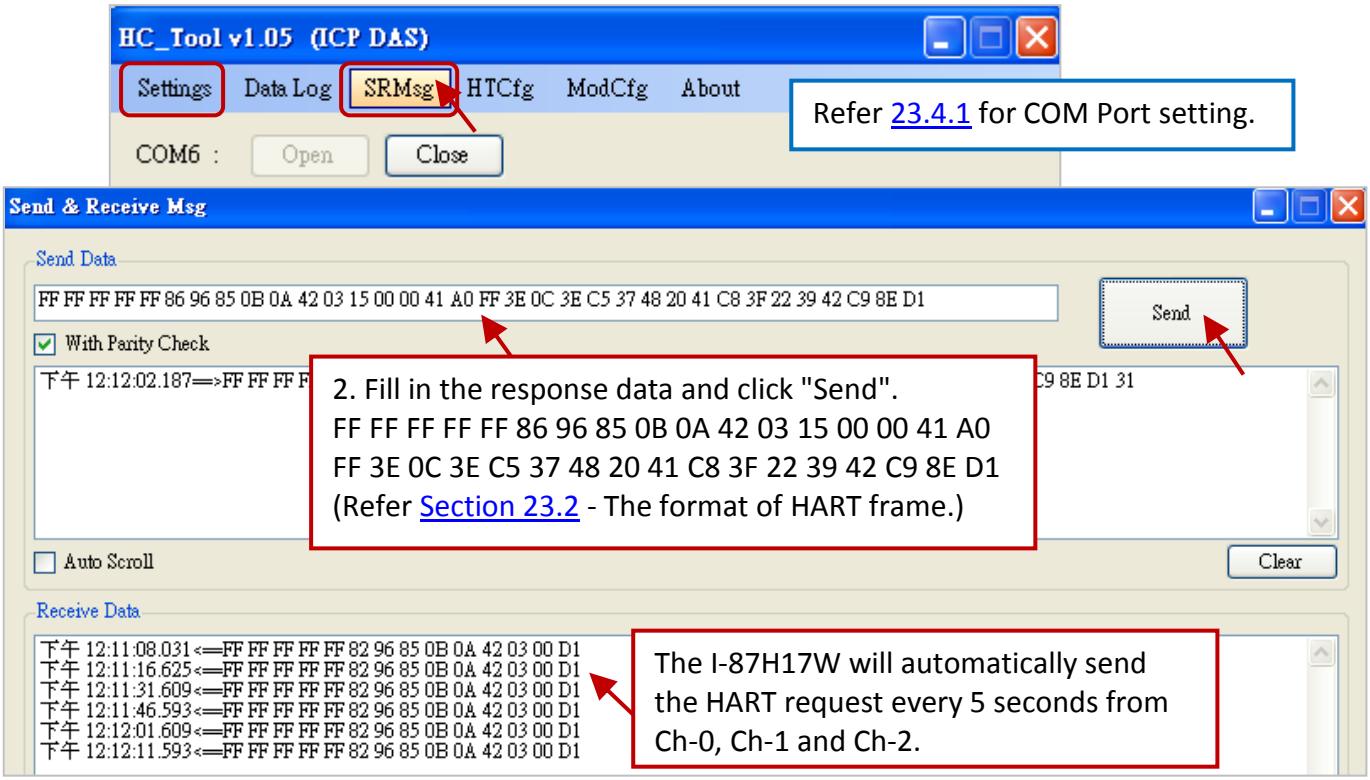
Feature Descriptions: (Program- Demo_HART_2 ; Wiring – See Section 23.4.1)

To send/receive the HART frame automatically and repeatedly every 5 seconds from the Ch-0, Ch-1 and Ch-2 of the I-87H17W module in the slot 2.

After downloading this project, open the spy list (NewSpy) and the I-87H17W will automatically send the HART request from Ch-0, Ch-1 and Ch-2.

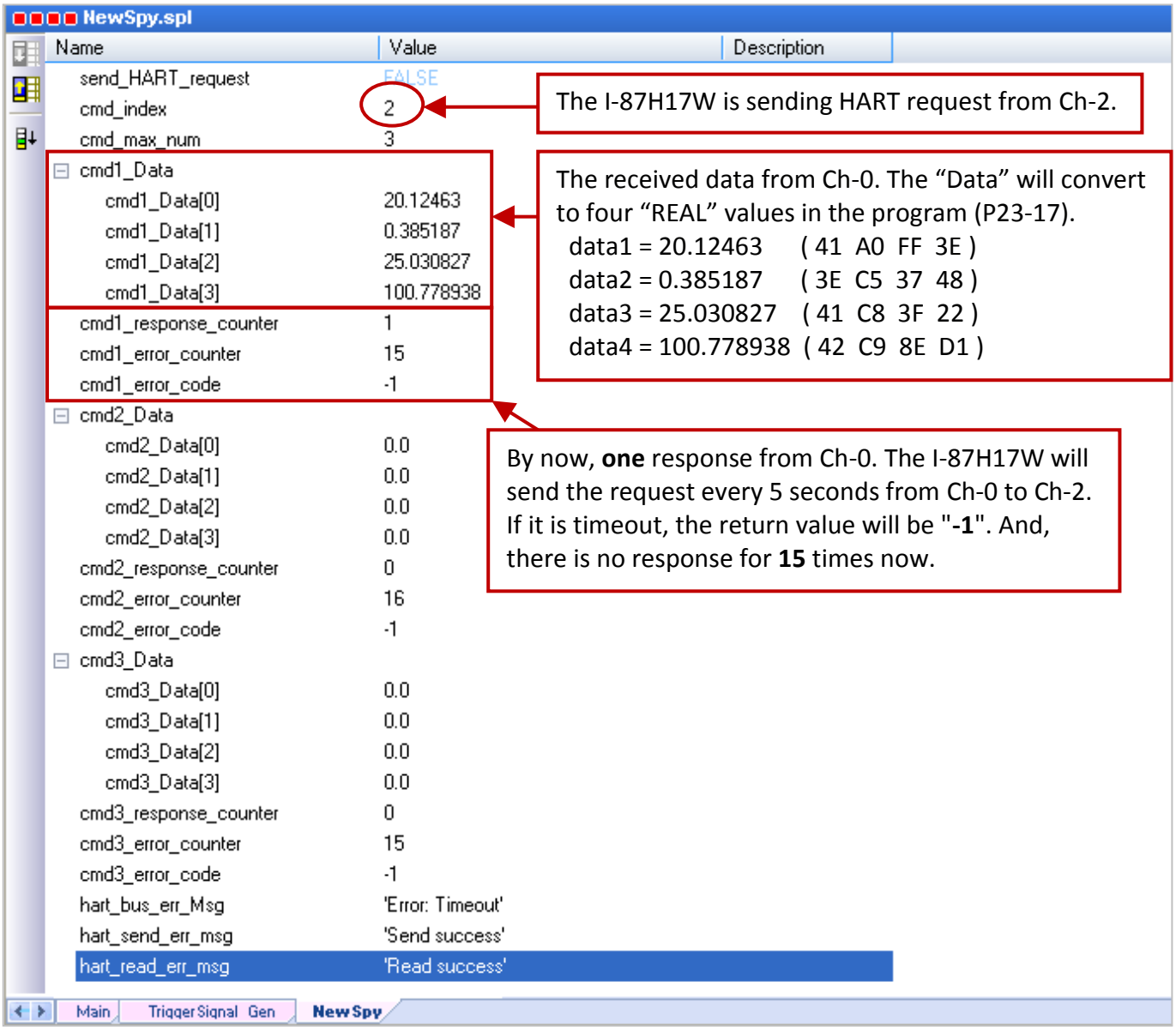


In the "HC_Tool", set up the COM Port and HART format first (refer 23.4.1), and then click "SRMsg" to open "Send & Receive Msg" window, fill in the response data and click "Send".



(It shows 15 seconds because we just link the channel 0 in this test.)

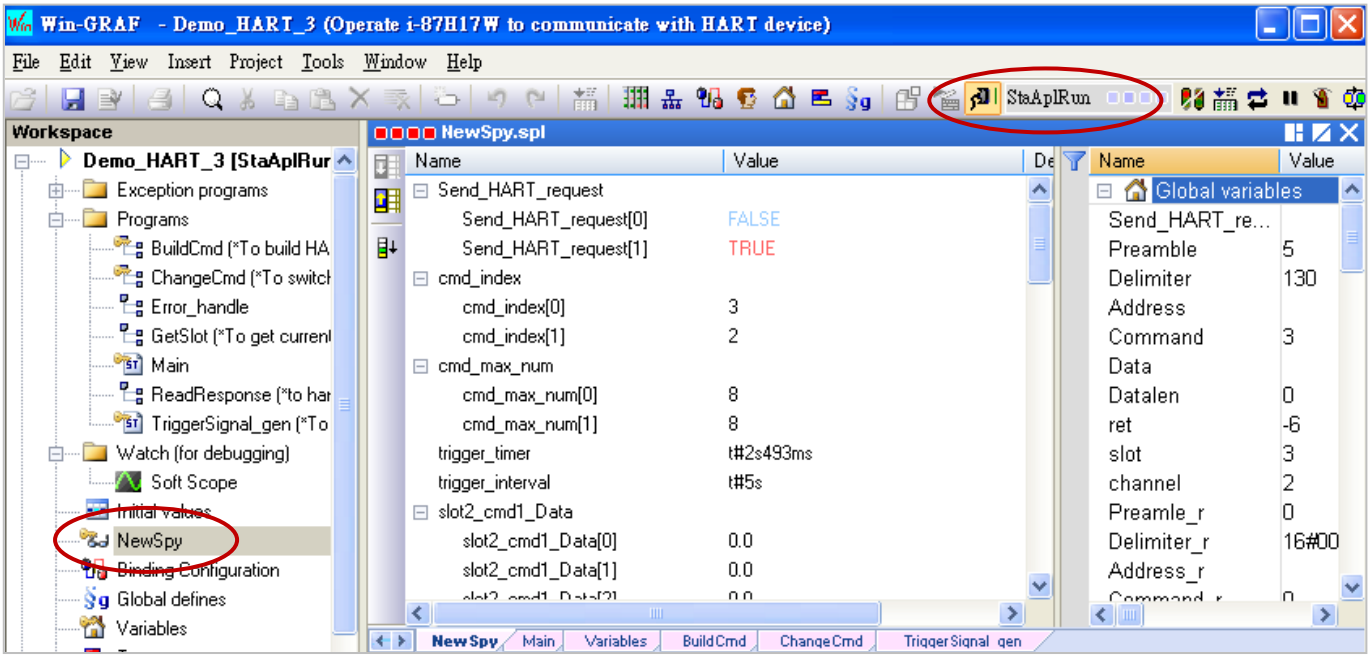
Preamble	Delimiter	Address	Command	Byte Count	Data
The HART request from the I-87H17W (Hex.)					
FF FF FF FF FF	82	(long) 96 85 0B 0A 42	03	00	-
The HART response from the device (Hex.)					
FF FF FF FF FF	86	(long) 96 85 0B 0A 42	03	$15_{(16)}$	00 00 41 A0 FF 3E 0C 3E C5 37 48 20 41 C8 3F 22 39 42 C9 8E D1



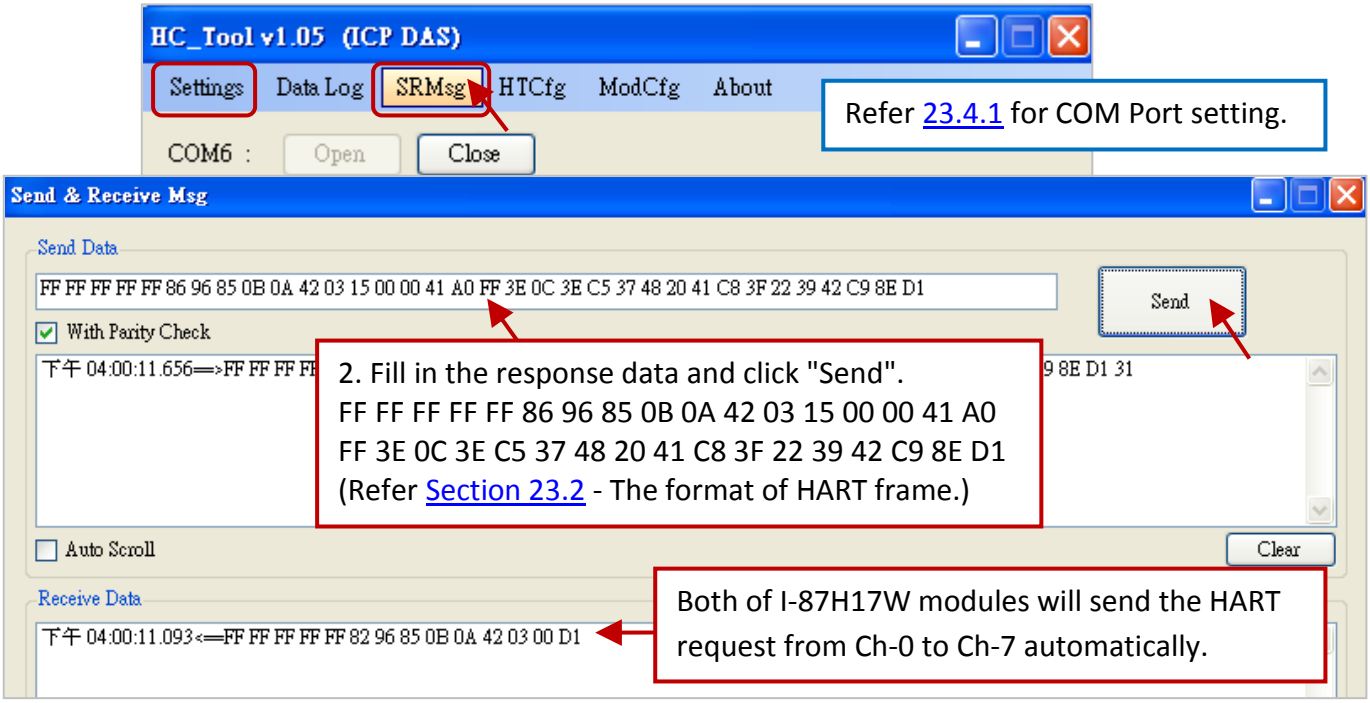
Feature Descriptions: (Program- Demo_HART_3 ; Wiring – See Section 23.4.1)

Using two I-87H17W in the slot 2 and slot 3 to send and receive the HART frame every 5 seconds automatically and repeatedly from Ch-0 to Ch-7 at the same time.

After downloading this project, open the spy list (NewSpy) and both of I-87H17W modules will send the HART request from Ch-0 to Ch-7 automatically.



In the "HC_Tool", set up the COM Port and HART format first (refer 23.4.1), and then click "SRMsg" to open "Send & Receive Msg" window, fill in the response data and click "Send".



Preamble	Delimiter	Address	Command	Byte Count	Data
The HART request from the I-87H17W (Hex.)					
FF FF FF FF FF	82	(long) 96 85 0B 0A 42	03	00	-

Preamble	Delimiter	Address	Command	Byte Count	Data
The HART response from the device (Hex.)					
FF FF FF FF FF	86	(long) 96 85 0B 0A 42	03	$15_{(16)}$	00 00 41 A0 FF 3E 0C 3E C5 37 48 20 41 C8 3F 22 39 42 C9 8E D1

NewSpy.spl

Name	Value	Description
Send_HART_request		
Send_HART_request[0]	FALSE	
Send_HART_request[1]	TRUE	
cmd_index		
cmd_index[0]	3	
cmd_index[1]	2	
cmd_max_num		
cmd_max_num[0]	8	
cmd_max_num[1]	8	
trigger_timer	t#2s116ms	
trigger_interval	t#5s	
slot2_cmd1_Data		
slot2_cmd1_Data[0]	20.12463	
slot2_cmd1_Data[1]	0.385187	
slot2_cmd1_Data[2]	25.030827	
slot2_cmd1_Data[3]	100.778938	
slot2_cmd2_Data		
slot2_cmd3_Data		
slot2_cmd4_Data		
slot2_cmd5_Data		
slot2_cmd6_Data		
slot2_cmd7_Data		
slot2_cmd8_Data		
slot2_cmd_response_counter		
slot2_cmd_response_counter[0]	1	
slot2_cmd_response_counter[1]	0	
slot2_cmd_response_counter[2]	0	
slot2_cmd_response_counter[3]	0	
slot2_cmd_response_counter[4]	0	
slot2_cmd_response_counter[5]	0	
slot2_cmd_response_counter[6]	0	
slot2_cmd_response_counter[7]	0	
slot3_cmd1_Data		
slot3_cmd2_Data		
slot3_cmd3_Data		
slot3_cmd4_Data		
slot3_cmd5_Data		
slot3_cmd6_Data		
slot3_cmd7_Data		
slot3_cmd8_Data		
slot3_cmd_response_counter		
slot3_cmd_error_counter		
slot3_cmd_error_code		
hart_bus_err_Msg	'Error: Timeout'	
hart_send_err_msg	'Send Error: i-87H17W is off-line'	
hart_read_err_msg	'Read success'	

New Spy | Main | Variables | Build Cmd | Change Cmd | Trigger Signal gen

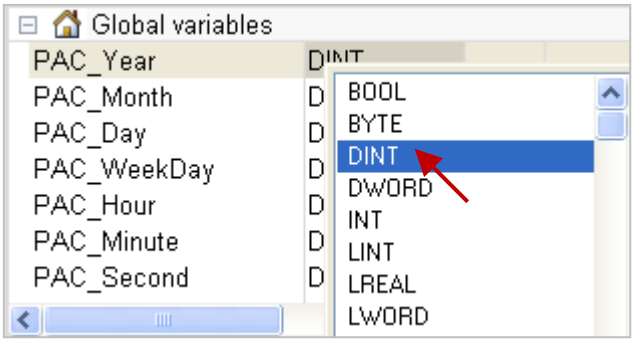
Send the HART request every 5 seconds.

The received data from Ch-0 of the I-87H17W in the Slot2. The "Data" will convert to four "REAL" values in the program.

data1 = 20.12463 (41 A0 FF 3E)
data2 = 0.385187 (3E C5 37 48)
data3 = 25.030827 (41 C8 3F 22)
data4 = 100.778938 (42 C9 8E D1)

Appendix A Data types and Ranges

Users can specify the data type of variables in the Variables Area (refer the Section 2.2.1) or in the Variables window (refer the Section 2.2.2).



Below are the available basic data types and ranges:

Data types	Size in-bits	Range of Values
BOOL (*)	---	TRUE, FALSE
SINT	8-bits (Small int, signed)	-128 to +127
USINT	8-bits (Unsigned small int)	0 to +255
BYTE		
INT	16-bits (Int, signed)	-32768 to +32767
UINT	16-bits (Unsigned int)	0 to +65535
WORD		
DINT (*)	32-bits (Double int, signed)	-2147483648 to +2147483647
UDINT	32-bits (Unsigned double int)	0 to +4294967295
DWORD		
LINT	64-bits (Large int, signed)	-2^{63} to $+(2^{63}-1)$
ULINT	64-bits (Unsigned large int)	0 to $+(2^{64}-1)$
LWORD		
Note: All the Win-GRAF PAC does not support the "ULINT" and "LWORD" data type.		
REAL (*)	32-bits (Floating point)	$\pm 3.4\times10^{-38}$ to $\pm 3.4\times10^{38}$
LREAL	64-bits (Floating point)	$\pm 1.7\times10^{-308}$ to $\pm 1.7\times10^{308}$
STRING (*)	A max. of 255 characters	---
TIME (*)	32-bits	T#0ms to T#23h59m59s999ms

(*): The commonly used data type.

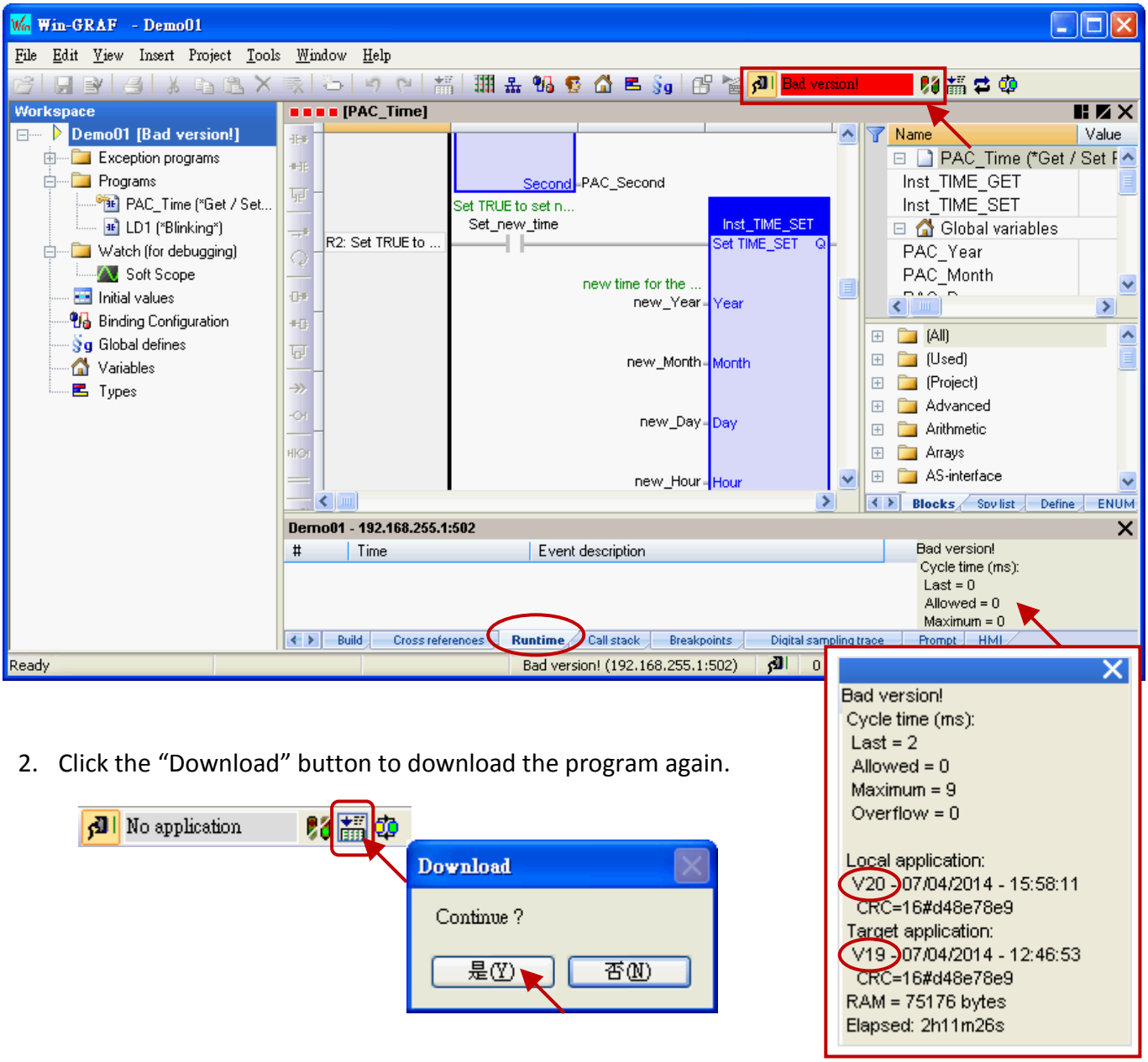
Appendix B Troubleshooting while On-Line the PAC

If the error message is showing up (as the screenshot below) after connecting to the Win-GRAF PAC, refer the following content to solve the problem.

- **The "Bad version!" error message:**
  It means that the compiled version between the PC and the PAC is different. The most common reason is that users have modified and re-compiled the program.

<u>To solve the problem</u>

1. Click the "Stop application" button to stop the running program.

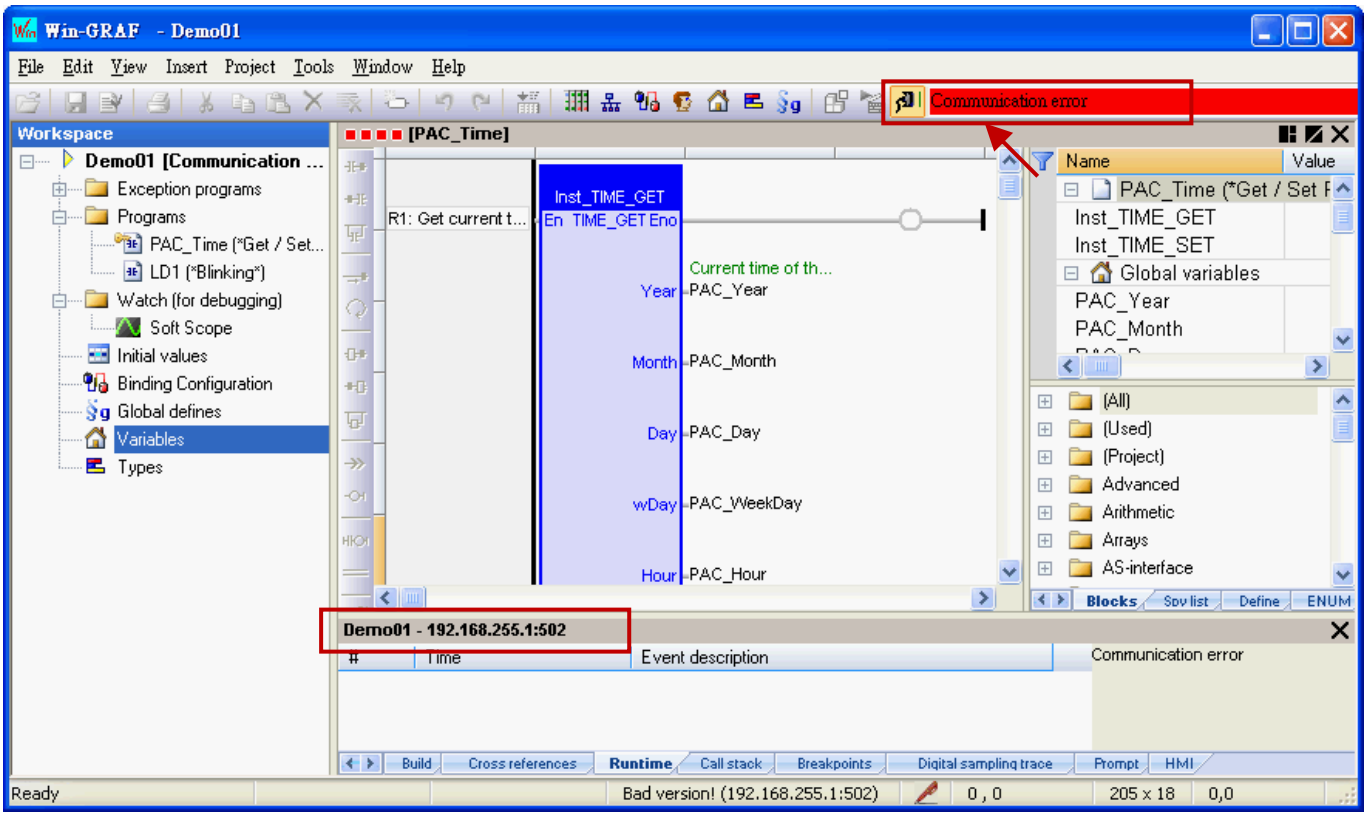


2. Click the "Download" button to download the program again.

3. The "RUN" message means that the program is working properly.

- **The "Communication error" error message:**

  A communications failure has occurred between the PC and the PAC.



<u>To solve the problem</u>

1. Make sure your Win-GRAF PAC is started, and the network communication between the PC and the PAC is functioning properly.
2. Make sure the IP setting of the Win-GRAF project is the same as the PAC IP (refer the Section 2.3.5, in this example, the IP address is "192.168.255.1:502").
3. Make sure the network communication of your PC is working.

Appendix C Enable the Screen Saver of WinCE PAC

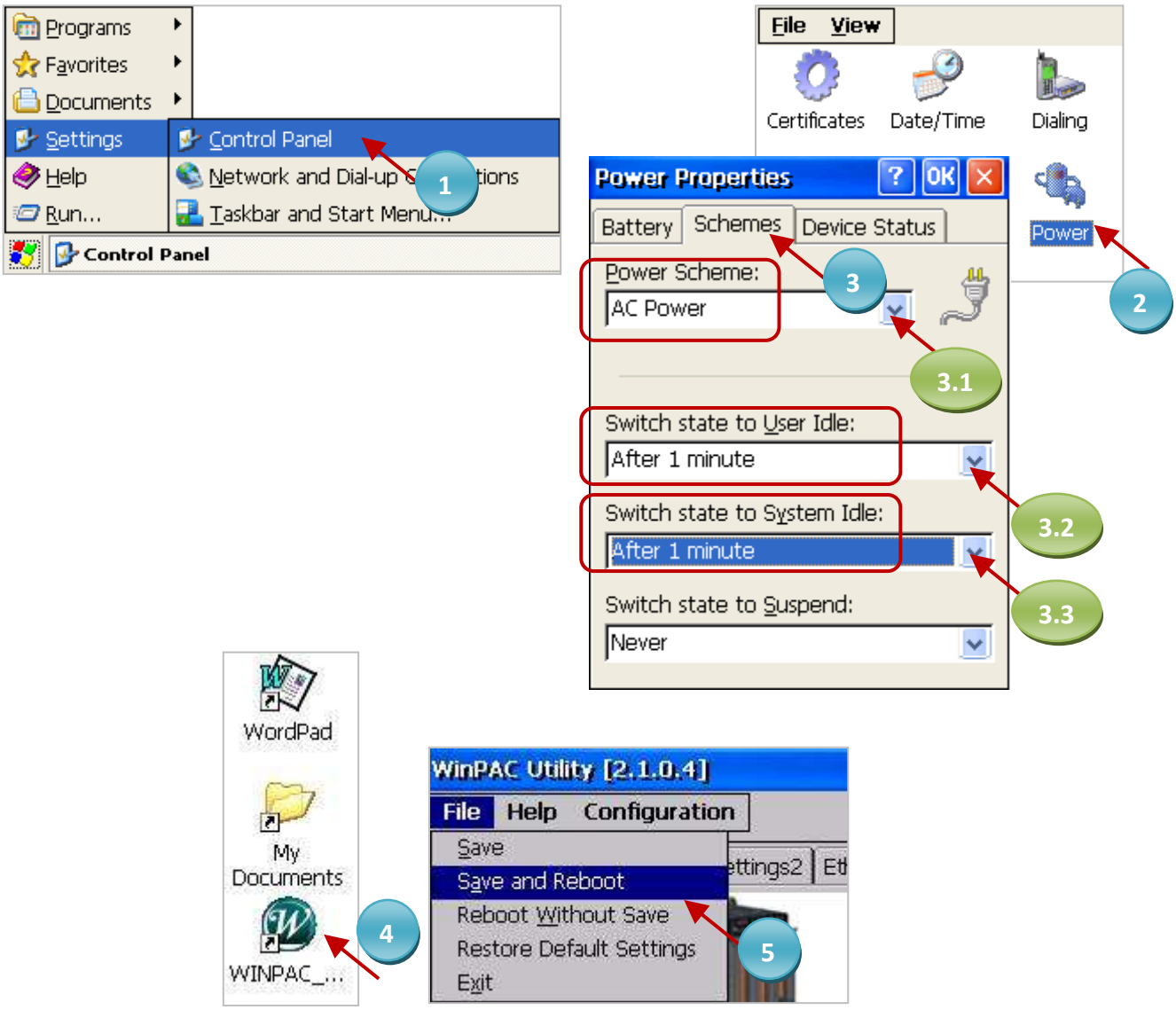
Please set the following two items to enable the screen of WinCE PAC.

1. Choose **"Control Panel**" > "Power" > "Schemes", and set the "Power Scheme" as "AC Power", set the "User Idle" and the "System Idle" to the same value (or set the "System Idle" value larger than the "User Idle" value).
2. Then, remember to run "**WinPAC Utility**" > "File" > "Save and Reboot" to save the settings and auto reboot the PAC.

<u>Using the WP-8xx8 as an example:</u>

If users do not touch the screen or button until the time out (e.g., 1 minute), the WP-8xx8 will turn off the backlight for enabling the screen saver. Whenever users touch the screen or button, the WP-8xx8 will turn on the backlight again.

The way to disable the screen saver is to set the "User Idle" and the "System Idle" as "Never", and then remember to run "WinPAC Utility" > "File" > "Save and Reboot" to save the settings and auto reboot the PAC.



Appendix D Using Expansion RS-232/485/422

The Win-GRAF PAC (See P1-1) expand more COM port in its slot No. 0 to 7 by using following modules.

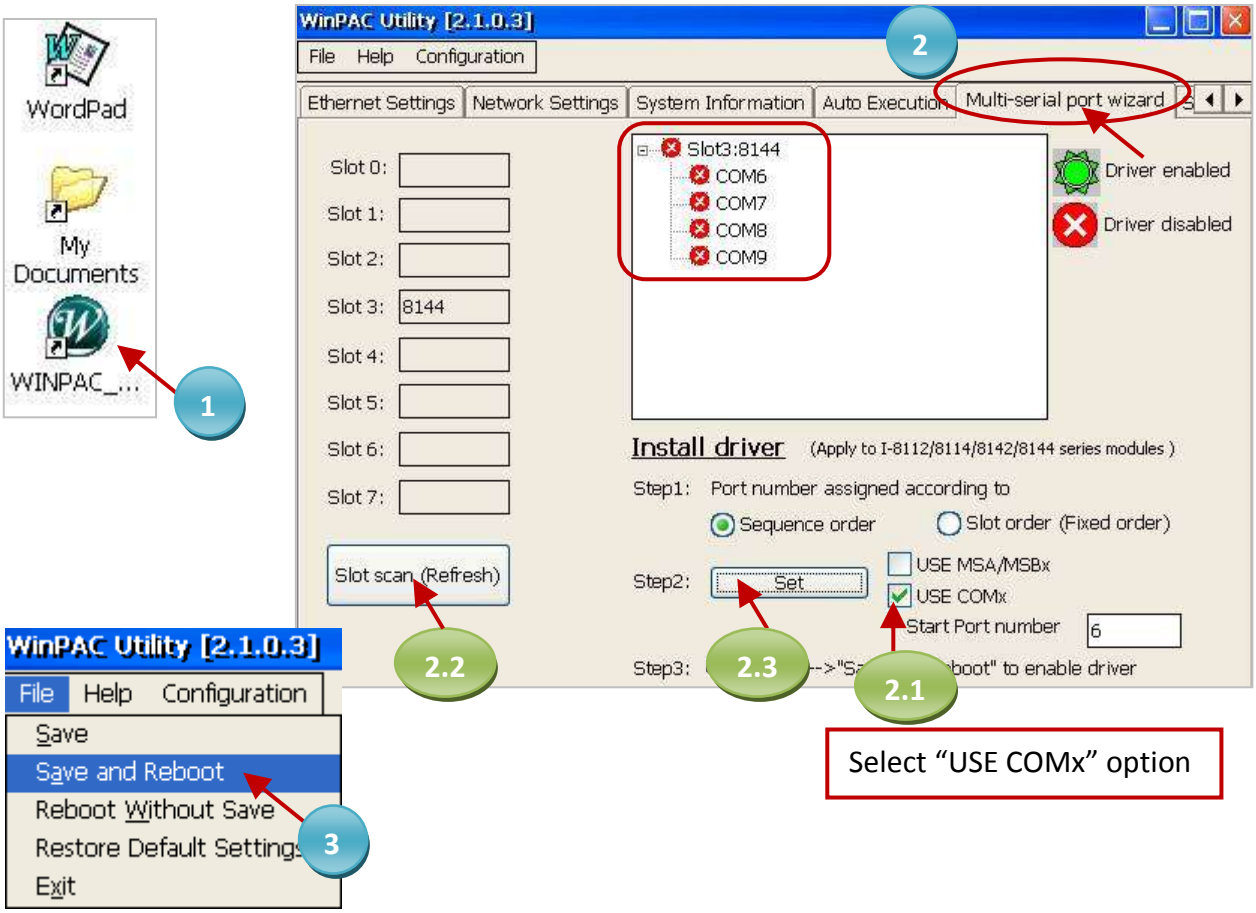
I-8112iW : 2-port Isolated RS-232 module
I-8114iW : 4-port Isolated RS-232 module
I-8114W : 4-port RS-232 module
I-8142iW : 2-port Isolated RS-422/RS-485 module
I-8144iW : 4-port Isolated RS-422/RS-485 module

Note: The **WP-5xx8** does not support **XW-5xx** series XW-board.
(This PAC can not expand COM port.)

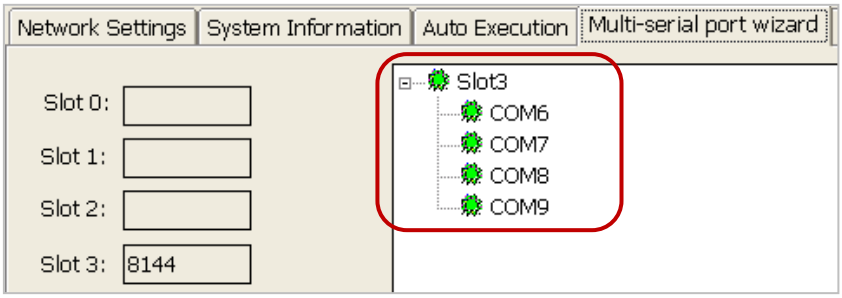
Using the WP-8xx8 as an example:

Before using these modules, please configure them by using the "WinPAC Utility". First, plug the module in the WP-8xx8's slot 0 to 7 (It is better to be in slot 0 to 3), and then run the "WinPAC Utility".

- Click the "Multi-serial port wizard" tab.
- Check the"USE COMx" option. (**Note:** The Win-GRAF doesn't support "USE MSA/MSBx")
- Click the "Slot scan" button, and then the current found serial-port expansion module will be listed on the left. (The earlier COM port setting is listed on the right if you have already set it before.)
- Click the "Set" button to refresh the new setting like the figure below.
- Click [File] > [Save and Reboot] to save the new setting and auto reboot the WP-8xx8.

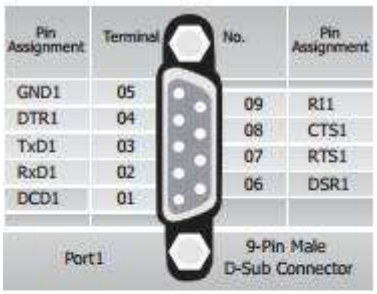


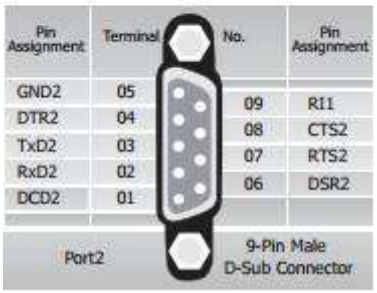
After the configuration succeeds, the COM port No. for the expansion is COM6 to COM37 in the Win-GRAF definition. (In this case, it expands the COM6 to COM9).



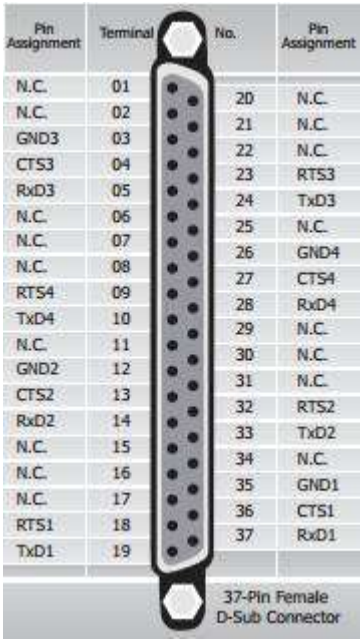
Pin Assignment:

I-8112iW
2-port Isolated RS-232

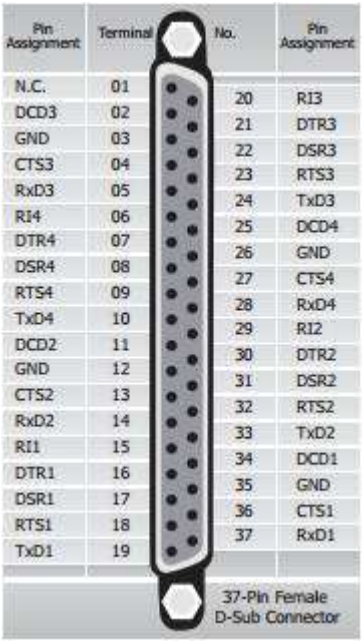




I-8114iW
4-port Isolated RS-232



I-8114W
4-port RS-232



I-8142iW

Terminal No.	Pin Assignment
01	D1+/TxD1+
02	D1-/TxD1-
03	RxD1+
04	RxD1-
05	GND1
06	D2+/TxD2+
07	D2-/TxD2-
08	RxD2+
09	RxD2-
10	GND2
11	N.C.
12	N.C.
13	N.C.
14	N.C.
15	N.C.
16	N.C.
17	N.C.
18	N.C.
19	N.C.
20	N.C.

I-8144iW

Terminal No.	Pin Assignment
01	D1+/TxD1+
02	D1-/TxD1-
03	RxD1+
04	RxD1-
05	GND1
06	D2+/TxD2+
07	D2-/TxD2-
08	RxD2+
09	RxD2-
10	GND2
11	D3+/TxD3+
12	D3-/TxD3-
13	RxD3+
14	RxD3-
15	GND3
16	D4+/TxD4+
17	D4-/TxD4-
18	RxD4+
19	RxD4-
20	GND4

I-8142iW (2-port Isolated RS-422/485)
RS-485 port1: (D1+ , D1-)
RS-485 port2: (D2+ , D2-)

RS-422 port1: (TxD1+ , TxD1-, RxD1+, RxD1-)
RS-422 port2: (TxD2+ , TxD2-, RxD2+, RxD2-)

I-8144iW (4-port Isolated RS-422/485)
RS-485 port1: (D1+ , D1-)
RS-485 port2: (D2+ , D2-)
RS-485 port3: (D3+ , D3-)
RS-485 port4: (D4+ , D4-)

RS-422 port1: (TxD1+ , TxD1-, RxD1+, RxD1-)
RS-422 port2: (TxD2+ , TxD2-, RxD2+, RxD2-)
RS-422 port3: (TxD3+ , TxD3-, RxD3+, RxD3-)
RS-422 port4: (TxD4+ , TxD4-, RxD4+, RxD4-)

Appendix E Enabling a Serial Port for Connecting the Win-GRAF Workbench

(In this section we use the WP-8xx8 as an example to show the way to enable the serial port for connecting the Win-GRAF Workbench and this way is also applied to other Win-GRAF PACs.)

The Win-GRAF PAC's Ethernet Port is typically enabled for the Win-GRAF Workbench to debug or download or upload the project. If users want to enable a serial port (i.e., RS-232 or RS-485) for doing these operations, follow the way as below:

Method 1:

When the Win-GRAF PAC is turned on, it will try to read a "Extra_Ports.txt" file in the path "\System_Disk\Win-GRAF\", and the contents are shown like below.

COM1:19200,N,8,1

It means to enable the COM1 and the Baud Rate is 19200 bps

If you want to enable the COM2 and its Baud Rate is 9600 bps, modify the contents as below.

COM2:9600,N,8,1

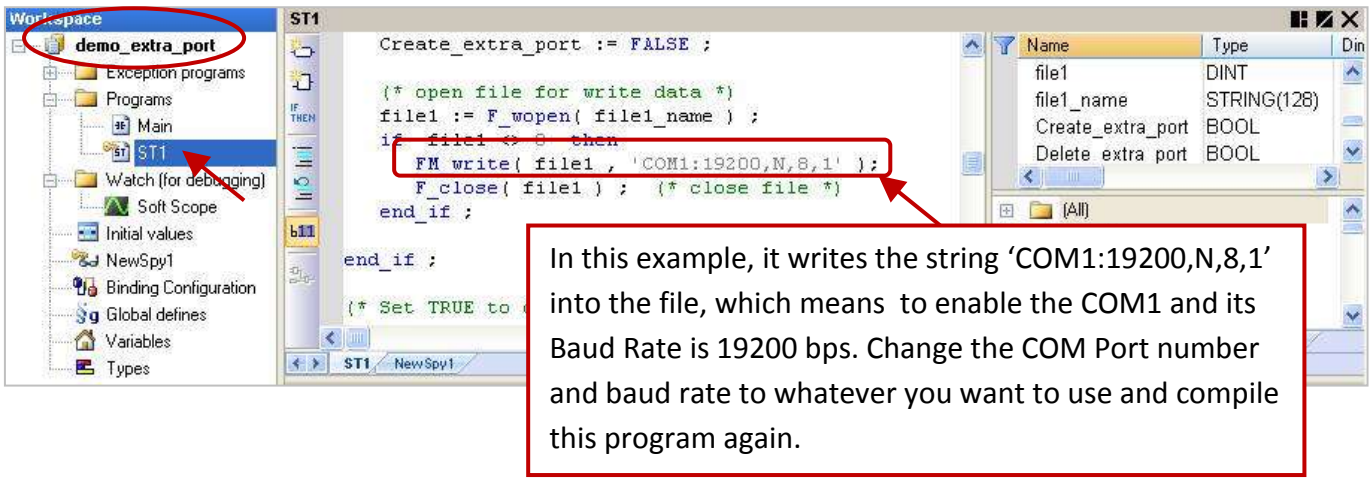
So, put this text file to the path "\System_Disk\Win-GRAF\" by FTP and then reboot the PAC.

Disable the COM Port:

If you want to cancel this COM Port setting, simply delete the "Extra_Ports.txt" file on the PAC (\System_Disk\Win-GRAF\Extra_Ports.txt) and then reboot the PAC.

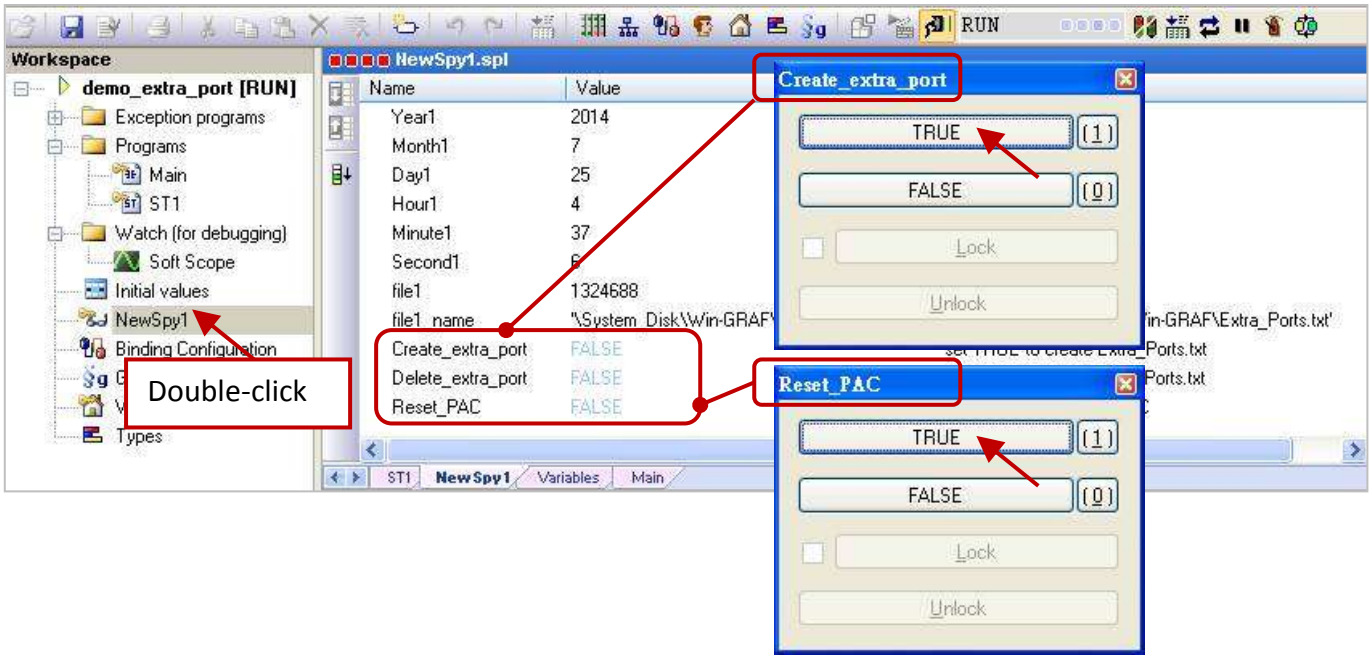
Method 2:

Users can find out a "demo_extra_port" demo project in the shipment CD. First, restore this zip file (refer the Chapter 12) and then mouse double click "ST1" to modify this program. Finally, compile and download this program to the PAC (refer the Section 2.3.5).

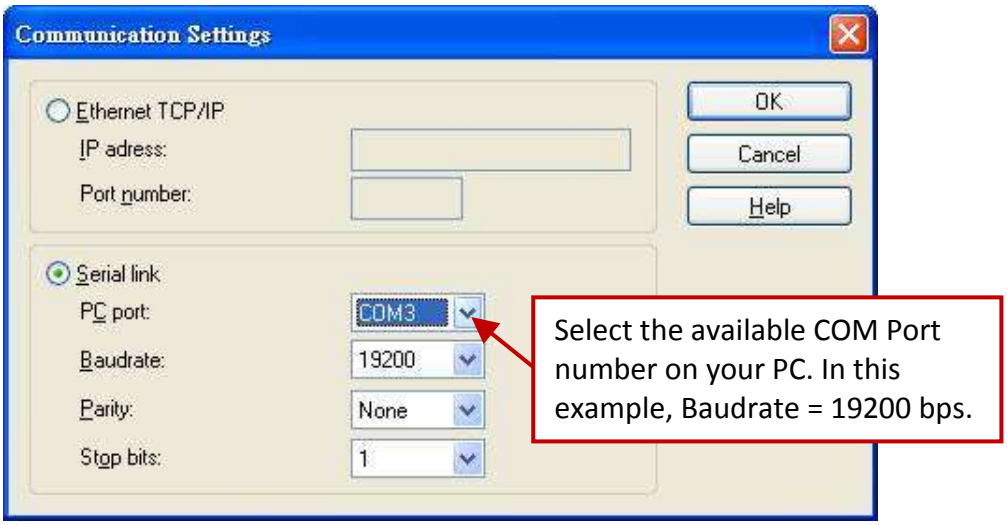


Test the program:

1. After connecting the Win-GRAF PAC, set the variable -"Create_extra_port" as "TRUE" in the Spy list. In this example, it will add a file -"Extra_Ports.txt" into the path "\System_Disk\Win-GRAF\" and the content is “COM1:19200,N,8,1”.
2. Set the"Reset_PAC" variable as "TRUE", and then the PAC will auto reboot and apply the setting.



In the Win-GRAF Workbench, if you want to connect to the Win-GRAF PAC through the serial port. Refer the Section 2.3.5 – Step 1 to 2, and select the "Serial link" option for connecting the PAC via COM Port.



Disable the COM Port:

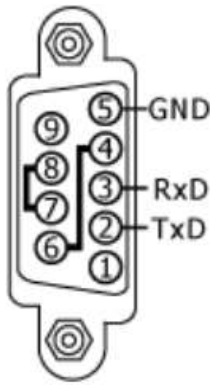
If you want to cancel this COM Port setting, simply set the "Delete_extra_port" variable as "TRUE", it will delete the “Extra_Ports.txt” in the path (PAC: \System_Disk\Win-GRAF\). Then, set the "Reset_PAC" variable as "TRUE" to auto reboot the PAC.

Appendix F Pin Assignment of PAC's Serial Ports

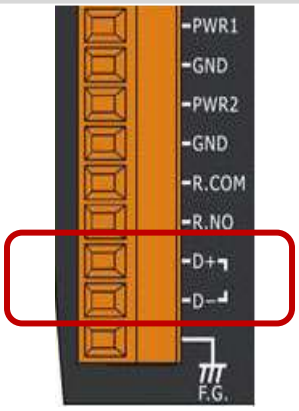
WP-8448/8848:

Pin assignment of COM1 to COM4.

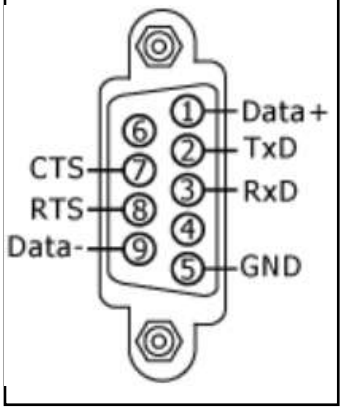
COM1: RS-232



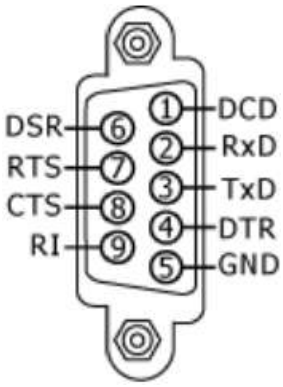
COM2: RS-485



COM3: RS-232/485



COM4: RS-232

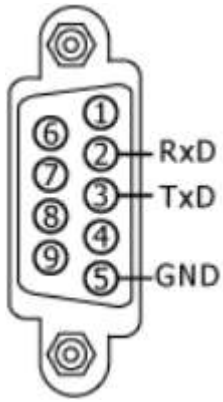


Note: WP-8148 has no COM3, COM4.

XP-8038-CE6/8138-CE6/8338-CE6/8738-CE6 and XP-8048-CE6/8348-CE6/8748-CE6:

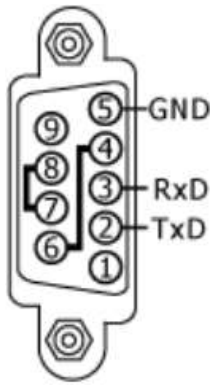
Pin assignment of COM1 to COM5.

COM1: RS-232

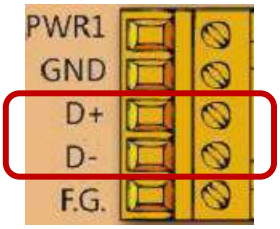


External COM1 for the **XP-8038-CE6** and the **XP-8048-CE6** only;
The COM1 of other PACs is for internal communications with I-87KW modules in slots only.

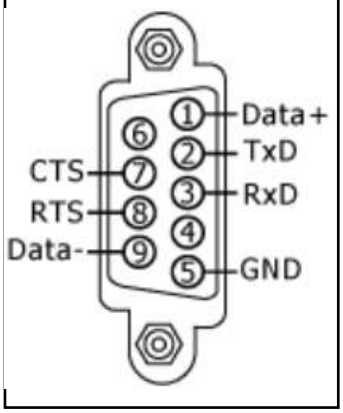
COM2: RS-232



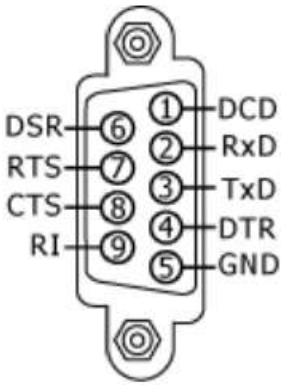
COM3: RS-485



COM4: RS-232/485

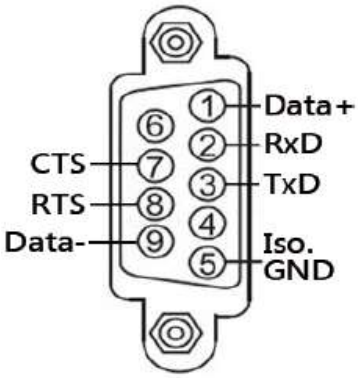
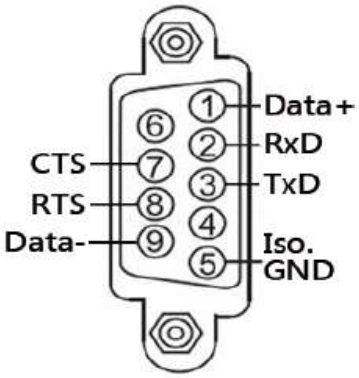
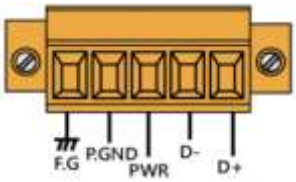


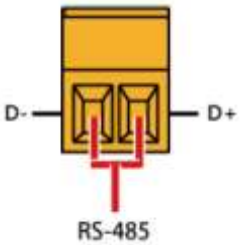
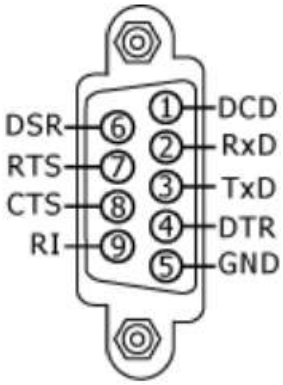
COM5: RS-232



VP-x2x8-CE7 **:**

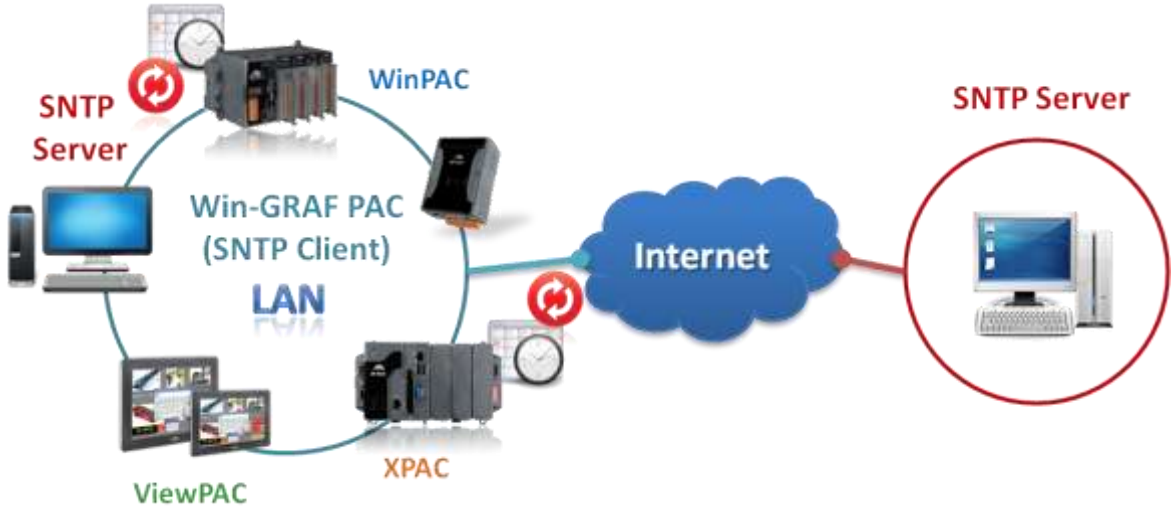
Pin assignment of COM1 to COM3.

VP-2208-CE7	VP-4208-CE7	
COM1 & COM2 : RS-232/RS-485	**COM1 & COM2 : RS-232/RS-485**	**COM3 : RS-485 (D+, D-)**
		

VP-1238-CE7	
COM2 : RS-485 (D2+, D2-)	**COM3 : RS-232**
	

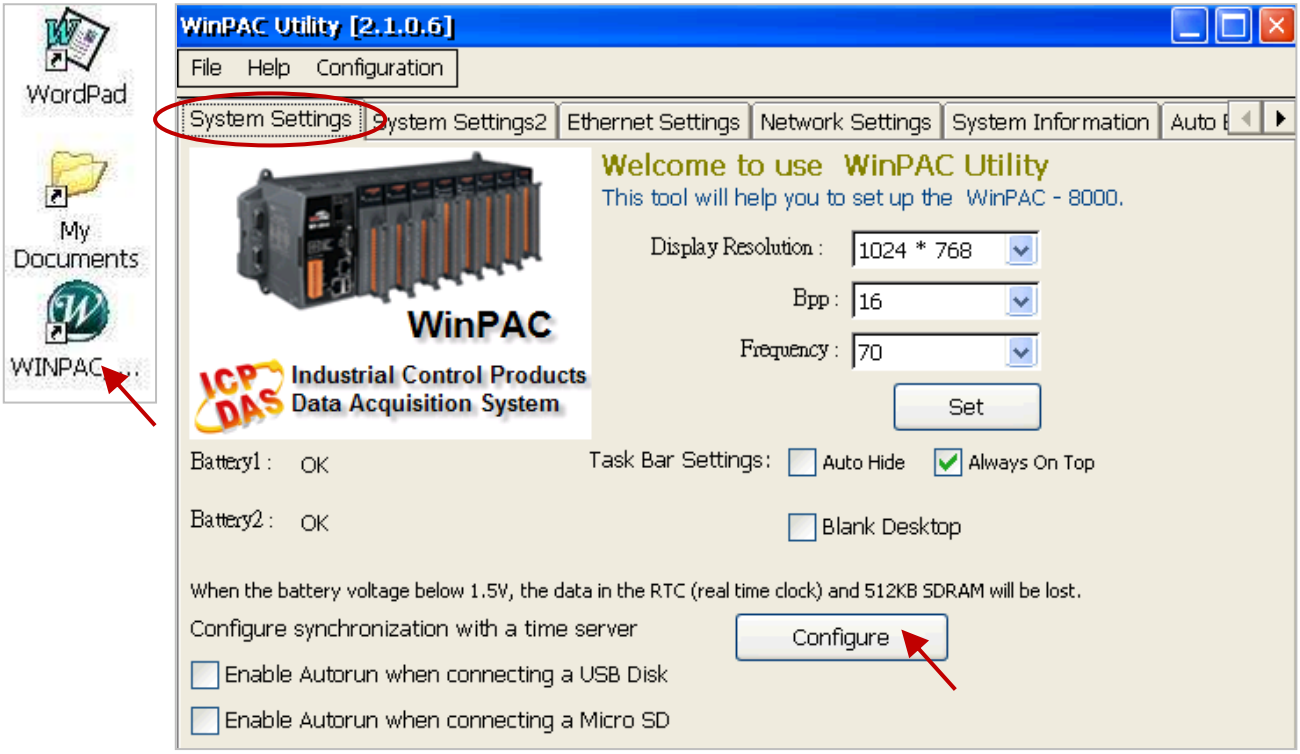
Appendix G Automatically and Periodically Synchronizing the PAC Time over a Network

The Win-GRAF PAC (WP-8xx8, WP-8xx8-CE7, WP-5238-CE7, XP-8xx8-CE6, VP-x208-CE7, and VP-x238-CE7) support SNTP (**S**imple **N**etwork **T**ime **P**rotocol) Client for network time synchronization. This chapter will describes how to synchronize the Win-GRAF PAC time with the SNTP Server over the Internet or a local network.



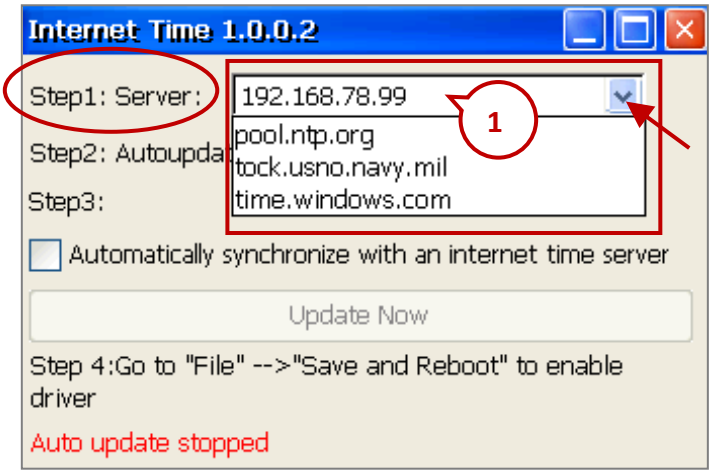
G.1 Set up an SNTP Client for Network Time Synchronization

Run the WinPAC Utility on the PAC's (e.g., WP-8xx8) desktop, select the "System Settings" tab and click the "Configure" button to open the "Internet Time" window.



Step 1 : Assign a SNTP Server

In the "Internet Time" window, you can select the listed SNTP Server (as the figure below) to conduct the Internet time synchronization. Or, you can set up a PC as a SNTP Server (See Section G.2) and then type its IP address (e.g., 192.168.78.99) in the "Server" field, for automatically time synchronization through the Internet or a local network.



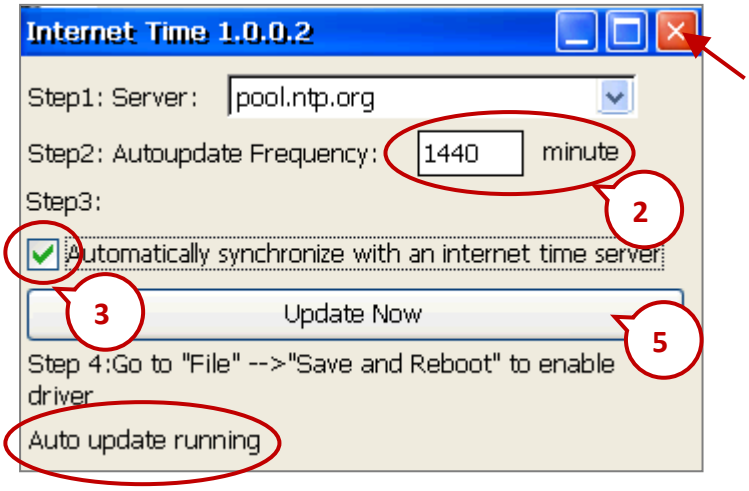
Note: If the assigned SNTP Server for the PAC time synchronization is over the Internet in a different domain, the user must set up the "Default Gateway" for the PAC (See Section G.3)

Step 2 : Assign the Auto-Update Frequency

If setting to update once a day (24 hours), entering "1440". (Uint: minutes. The minimum update frequency is "5" minutes.)

Step 3 : Enable Auto-Time-Synchronization

Check the box for time synchronization automatically with a SNTP Server, and then click "X" on the upper-right corner to exit this window. (Uncheck the box to stop this function.)



Step 4 : Save and reboot the PAC

Click "File" > "Save and Reboot" of the WinPAC Utility to save the settings and restart the PAC.

Step 5 : Test the SNTP Automatic Time Sync

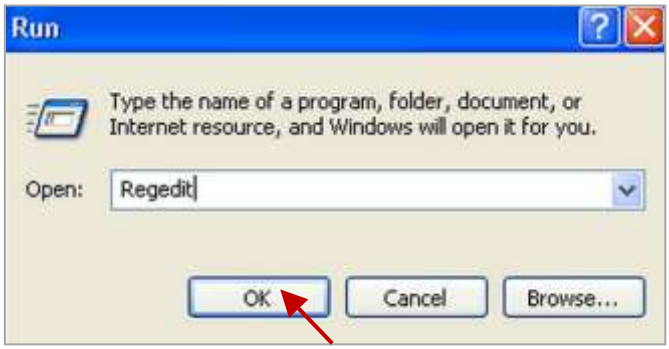
After rebooting the PAC and able to access the network, it will automatically synchronize the time with the SNTP Server according to the previous settings. You can also click the "Update Now" button in the "Internet Time" window to update the PAC time immediately.

G.2 Set up a Windows XP PC as the SNTP Server to test the SNTP Client

Prepare a Windows XP PC for the WinPAC SNTP Client to synchronize the time. Follow these steps.

Step 1 : Enable and Set Up the SNTP Server of Windows XP PC

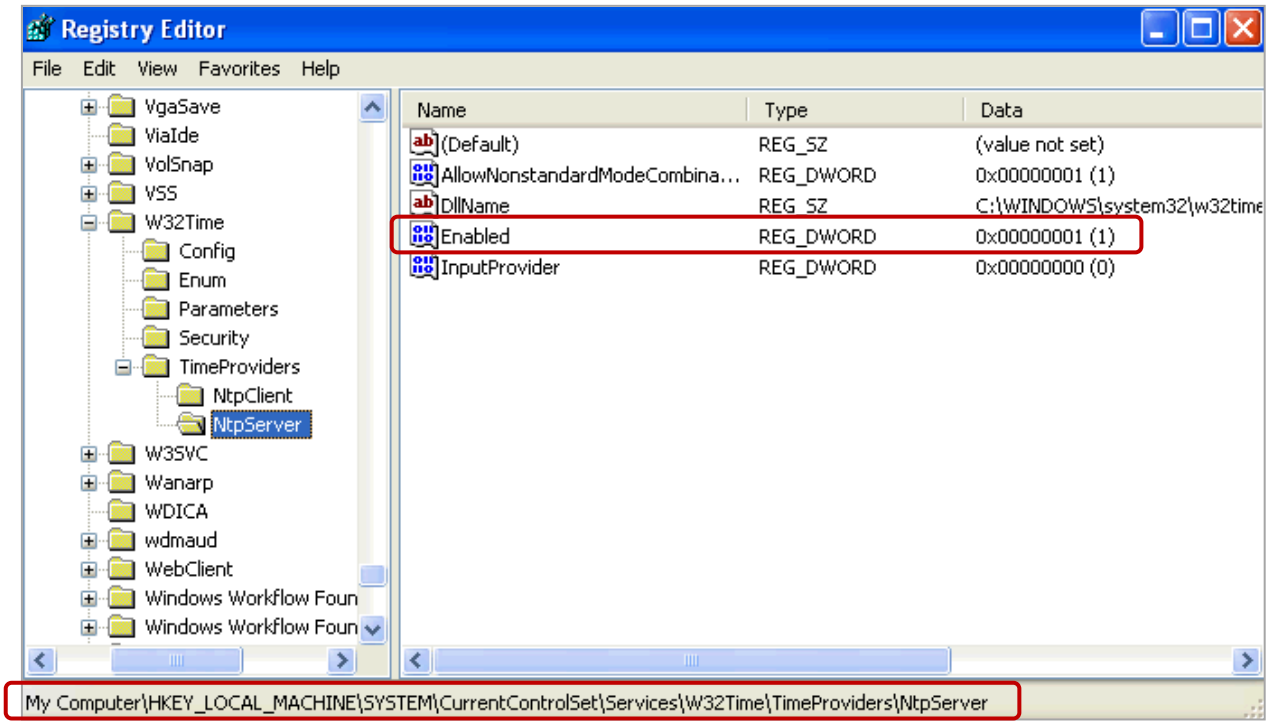
1. Run "Registry Editor"
   On PC, select [Start] > [Run] and enter "regedit", then click "OK".



2. Enable the SNTP Server.
   The SNTP Server in a normal Windows XP PC is default enabled.

   Left window:
   Please change to the following directory.
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time**TimeProviders\NtpServer**

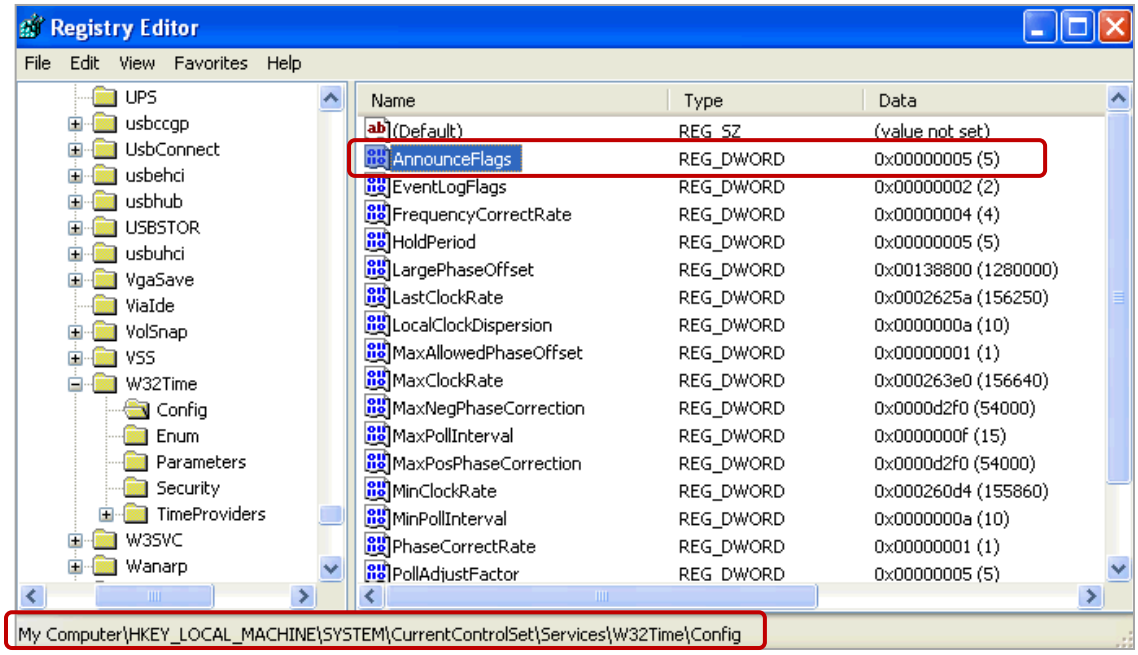
   Right window:
   The most right "Data" number of **"Enabled"** is (1) which means the SNTP Server is enabled. If it is (0) which means the NTP Server is disabled. Please right click "Enabled", select "Modify" and change "Value" to **"1"**, and then click "OK".



3. Set the Windows Time service to use the internal hardware clock.

   Left window:
   Please change to the following directory.
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time**Config**

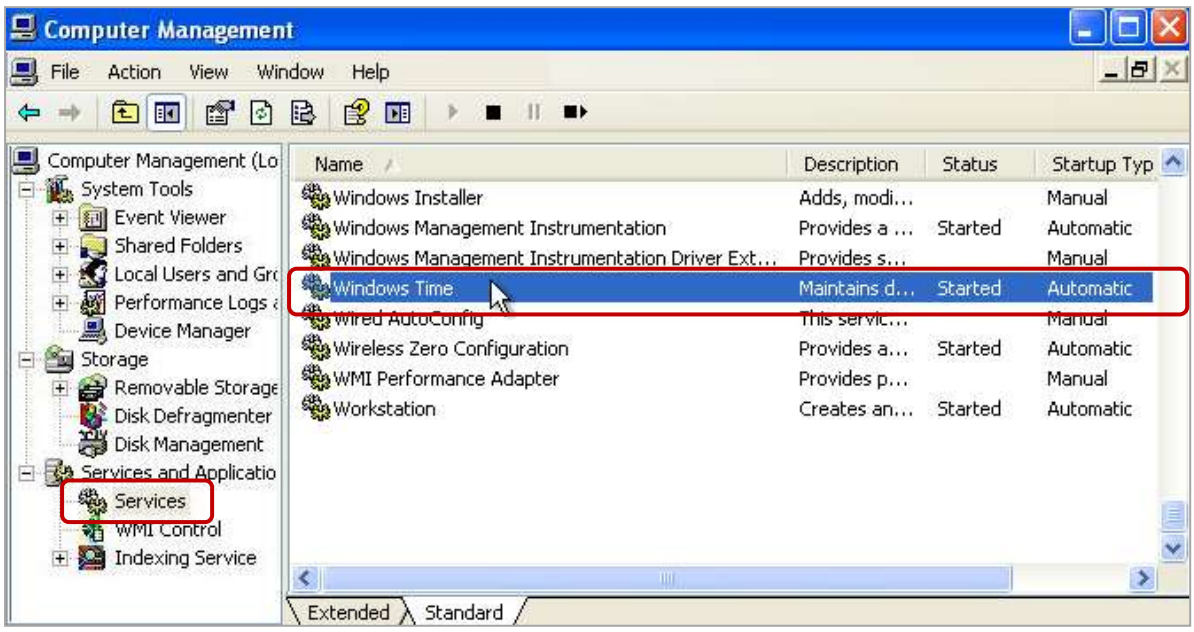
   Right window:
   Right click **"AnnounceFlags"**, select "Modify" and change "Value" to **"5"**. Then, click "OK" to exit the "Registry Editor" window.



Step 2 : Restart Windows Time Service

1. On Windows XP desktop, click [Start] > [Run].
2. Enter "**net stop w32time && net start w32time**", and then click "OK".

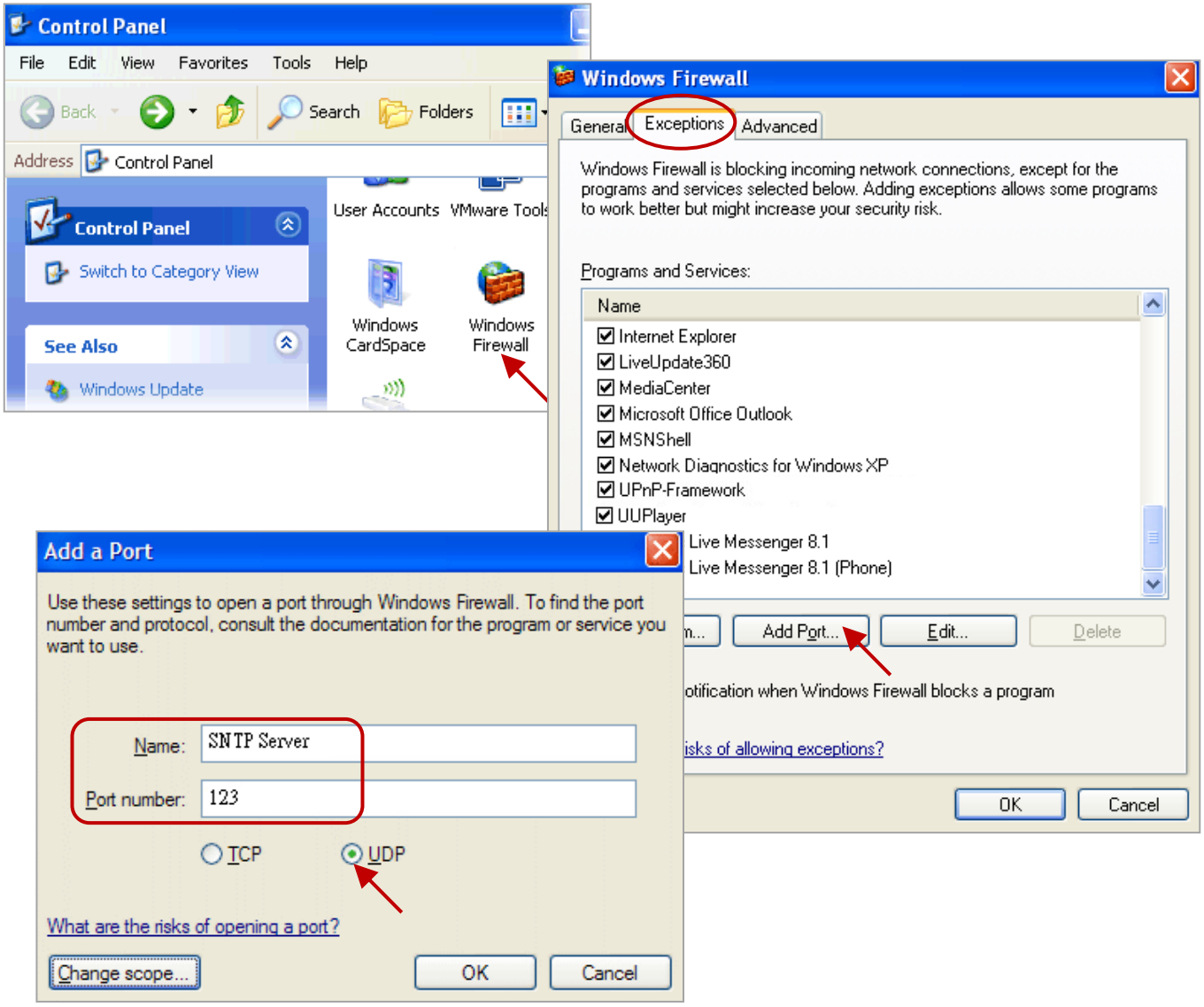
The user can also right click [My Computer], left click [Manage] and [Service], and double-click [Windows Time] to stop/start the Windows Time service and set its "Startup Type" to "Automatic".



Step 3 : About the Windows Firewall

If using Windows Firewall (enabled), you need to open a port (i.e., UDP123) for allowing communicates with other device.

1. On Windows XP desktop, click [Start] > [Control Panel] > [Windows Firewall].
2. Click the “Exceptions” tab, and click “Add Port…”.
3. Give a name “SNTP Server”, set up the port number “123” and select “UDP”. Click “OK” to exit.



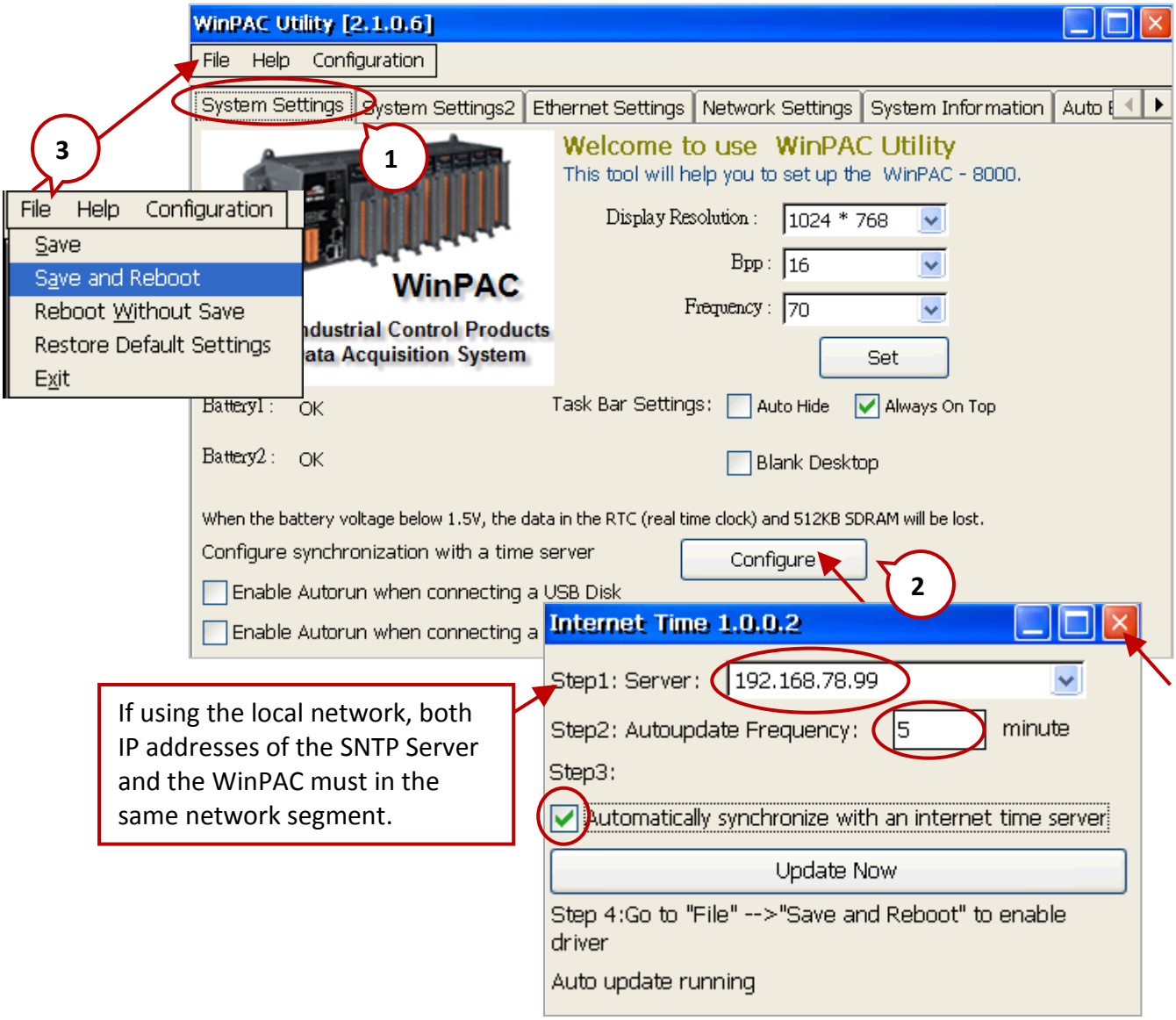
Step 4 : Set up a fixed IP for the SNTP Server

The SNTP Server must set up a static IP address for the time synchronization service no matter over the Internet or over a local network. And, this IP address must the same with the settings of the WinPAC Utility (See Section G.1). If using the local network, both IP addresses of the SNTP Server and the WinPAC must in the same network segment so that it can sucessfully synchronize the PAC time.

For example, the IP/Mask addresses of the WinPAC are "**192.168**.80.21 / **255.255**.0.0" and the IP/Mask addresses of the SNTP Server are "**192.168**.78.99 / **255.255**.0.0". They are in the same network segment.

Step 5 : Testing

After setting up the SNTP Server, the user can test the SNTP Client (e.g., the WinPAC). Follow the instructions in Section G.1 or see the figure below, type the IP address of the SNTP Server, change the auto-update frequency to "5" minutes, checked the auto-time-synchronization box, and reboot the Win-GRAF PAC (e.g., WP-8xx8).



You can double click the taskbar to see if the date/time is synchronized automatically. Then, change the auto-update frequency to the needed value.

Server

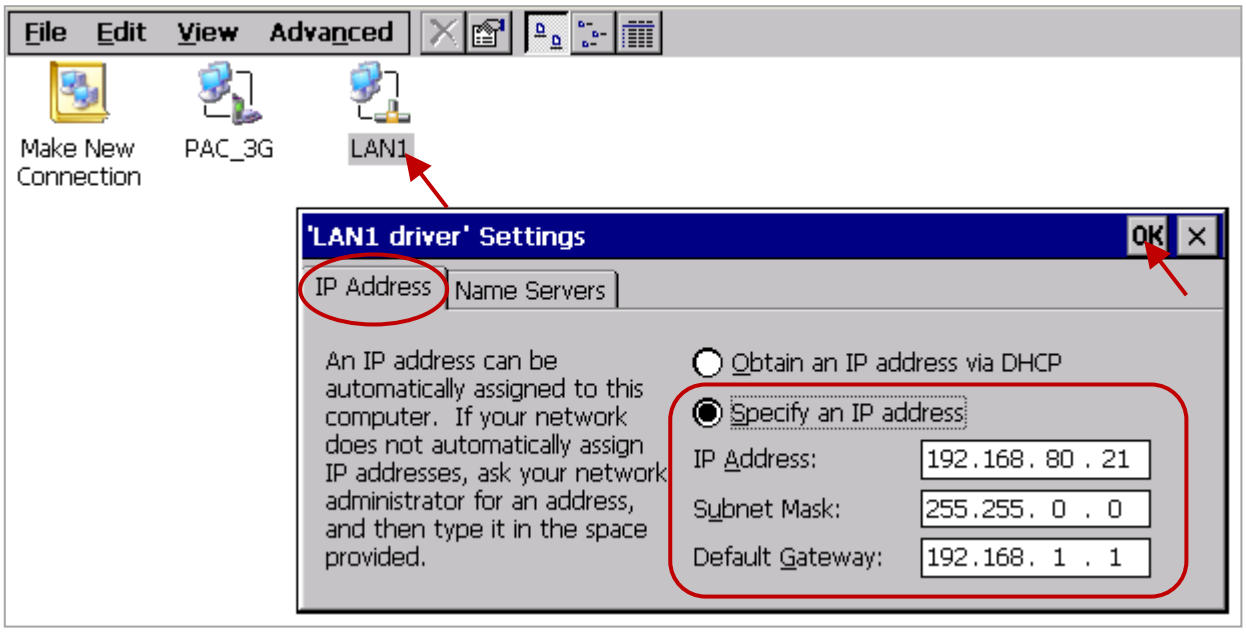
PAC

G.3 Set up the Gateway and DNS Server Addresses for the PAC

If your PAC (i.e., SNTP Client) synchronize the time with the SNTP Server over the Internet (i.e., in two different networks), the user not only to set up the IP and Mask addresses, but to set up the Gateway and DNS Server addresses.

How-to: (Using the WinPAC as an example)

1. On the PAC desktop, click [Start] > [Settings] > [Network and Dual-up Connections], and then double click on LAN1 (or LAN2). (See Section 1.3 if not familiar with the operations.)
2. Enter the Gateway address (e.g., "192.168.1.1") according to your application needs.
   Note: The Win-GRAF application **must use the fixed IP address,** no DHCP accepted.



3. Click "Name Server" tab to set up "Primary DNS". After completing it, click "OK" and reboot the PAC. (The Google Public DNS IP address is "8.8.8.8" and the Hinet DNS Server IP address is "168.95.1.1". So, you can choose one of them or enter a proper IP address.)